

# isCOBOL Evolve: Release Overview

---



## **Copyrights**

Copyright (c) 2024 Veryant  
6390 Greenwich Drive, #225, San Diego, CA 92122, USA

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.

# Table of Contents

Release Information .....	1
isCOBOL EIS (Enterprise Information System) .....	1
isCOBOL IDE Enhancements .....	2
Compiler Enhancements .....	9
Debugger Improvements .....	12
isCOBOL DataBase Bridge Improvements .....	13
New User Interface Features .....	14
Framework Improvements .....	16
Other enhancements in isCOBOL 2014R1 .....	18
isCOBOL Server Improvements .....	18
Web Direct 2.0 Improvements .....	18



## isCOBOL 2024 Release 1 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2024 R1.

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

The new version supports the latest Java LTS release 21.

isCOBOL IDE is now based on a newer Eclipse version, 2023-09 (4.29).

isCOBOL Evolve 2024 R1 supports Jakarta in all applications running in web environments.

Starting from this release, it's easier to port an application written for a specific RDBMS to another without changing source code.

Details on these enhancements and updates are included below.

### Java 21 support

isCOBOL Evolve and isCOBOL SDK 2024R1 are certified for Java 21, the current Java LTS (Long Term Support) release. It is important to have isCOBOL Evolve 2024R1 qualified to be used in the latest LTS version of Java because LTS applies the tenets of reliability engineering to the software development process and software release life cycle. Long-term support extends the period of software maintenance; it also alters the type and frequency of software updates (patches) to reduce the risk, expense, and disruption of software deployment, while promoting the dependability of the software.

We at Veryant feel it's important to test and qualify our products for Java LTS releases, and our internal tests are now performed on all 4 Java LTS versions: 1.8, 11, 17 and 21 to ensure the correct execution on each. Our users can then decide if and when to upgrade to a specific Java version, depending on their needs. Typically, COBOL programs function seamlessly across various Java versions. However, if these programs require interaction with third-party Java classes or libraries, it may become essential to utilize a specific version of the Java runtime. Having our products certified for any Java LTS version ensures that our users can move freely to the best available option.

One of the changes related to supporting Java 21 is the default file encoding, which is now UTF-8 on Windows platforms. If a COBOL application needs to support a different file encoding, it can be forced using the flag -Dfile.encoding startup option.

## isCOBOL IDE enhancements

The isCOBOL IDE 2024 R1 is now based on Eclipse 2023-09 (4.29), which includes full support for Java 21. The new version offers many improvements in performance, stability and usability on a day-by-day usage. The installers include an embedded JRE for the IDE, thus requiring no additional downloads from the developer.

Below is an extract of the new functionalities:

The Text Editors support multiple selections that allow most edit operations (text replacement or insertion, extend selection to next word or next line, copy / paste,...) to apply simultaneously on all ranges.

Multiple selection can be enabled by:

- Turning a block selection into multi-selection using the “To multi-selection” command
- Adding a caret with Alt-click.
- Using the new “Select All” button on the Find/Replace dialog.

The windows title bar in the dark theme on Microsoft Windows is now styled in the default dark theme.

As of Java 21 the IDE uses the UTF-8 default file encoding on Windows platforms as well as the previously supported platforms. If a COBOL project needs to support a different file encoding, it can be changed in the project properties.

## Jakarta support

Jakarta EE, formerly Java Platform, Enterprise Edition (Java EE) and Java 2 Platform, Enterprise Edition (J2EE), is a set of specifications extending Java SE with specifications for enterprise features such as distributed computing and web services. Jakarta EE applications are run on reference runtimes, which can be microservices or application servers. These handle transactions, security, scalability, concurrency and management of the components they are deploying.

JEE refers to Java servlet version 4, while Jakarta refers to Java servlet version starting from 5, and the class package name changes from javax to jakarta.

To run the newer servlet version 5, you need a newer Java container. This means Tomcat up to version 9 supports servlet version 4 while Tomcat 10 supports servlet version 5. Also, GlassFish 5 supports up to servlet version 4 while GlassFish 6 extends servlet support to version 5.

Veryant products now support servlet version 5 to allow deploying applications to the latest versions of Java Containers such as Tomcat and GlassFish.

isCOBOL 2024 R1 supports Jakarta in all our products that run in a web environment, such isCOBOL WebClient, isCOBOL EIS Servlets, WebServices and WebDirect.

### IsCOBOL WebClient

WebClient supports the Jakarta specification when it is run as a web application inside a Java Container, and will automatically activate the needed components in the weclient-server.war and webclient-admin-server.war files accordingly.

When WebClient is executed with the provided wrappers, it will use the embedded Jetty server.

### IsCOBOL EIS

The new Jakarta servlet specification is supported in isCOBOL EIS Servlets and WebServices when using the standard “web.xml” file available with the installation.

If you need to deploy on an older version of a Java Container, a file called “web.jee.servlet.xml” is provided that contains the entries needed, and is the same as in previous releases.

The main difference between the two files are the lines that refer the class name. The following lines are needed to enable JEE support:

```
...
<filter>
    <filter-name>isCOBOL filter</filter-name>
    <filter-class>com.iscobol.web.IscobolFilter</filter-class>
</filter>
...
<servlet>
    <servlet-name>isCobol</servlet-name>
    <servlet-class>com.iscobol.web.IscobolServletCall</servlet-class>
</servlet>
...
<listener>
    <listener-class>com.iscobol.web.IscobolSessionListener</listener-class>
</listener>
...
```

The following lines are needed to support the new Jakarta specification:

```
...
<filter>
    <filter-name>isCOBOL filter</filter-name>
    <filter-class>com.iscobol.webjakarta.IscobolFilter</filter-class>
</filter>
...
<servlet>
    <servlet-name>isCobol</servlet-name>
    <servlet-class>com.iscobol.webjakarta.IscobolServletCall</servlet-
class>
</servlet>
...
<listener>
    <listener-class>com.iscobol.webjakarta.IscobolSessionListener</listener-
class>
</listener>
...
```

No other changes are needed in other deployed libraries, since `iscobol.jar` contains all the necessary classes.

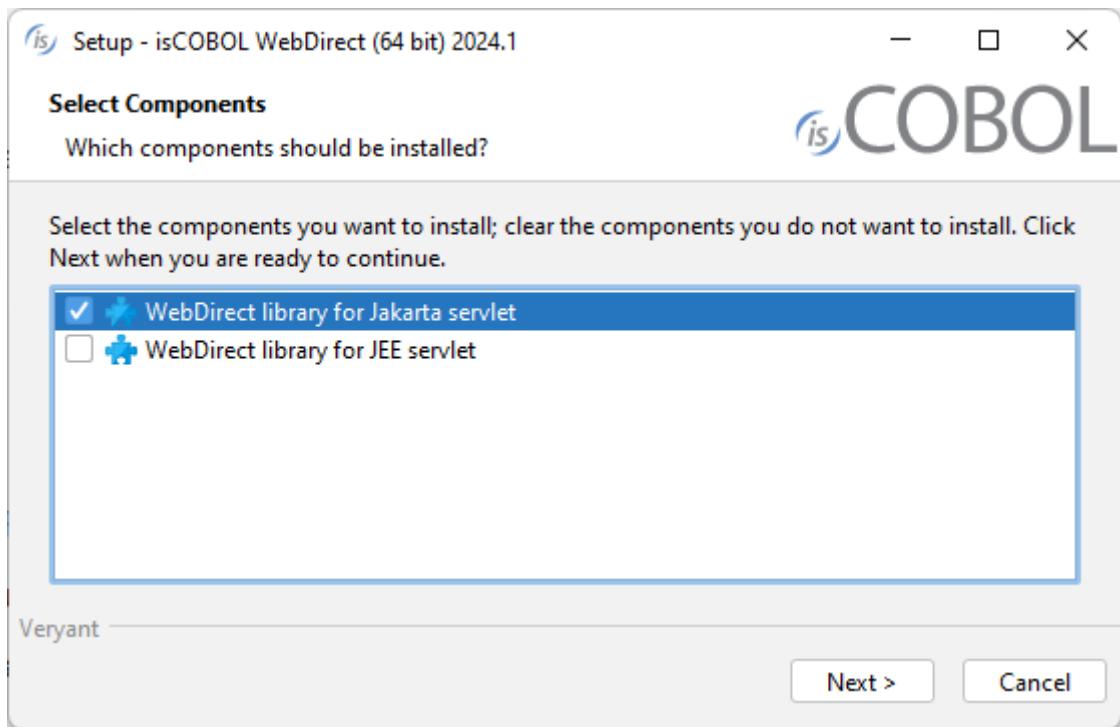
## IsCOBOL WebDirect

To support both JEE and Jakarta containers, two sets of libraries are provided for IsCOBOL EIS WebDirect deployment that are needed by the underlying ZK libraries on which WebDirect is built. IsCOBOL 2024 R1 provides a separate WebDirect installer that allows you to choose the set of libraries to install based on the container specification. The installer will deploy the selected libraries in the IsCOBOL SDK main folder.

If both JEE and Jakarta libraries are chosen, the installer creates two additional subfolders in the IsCOBOL SDK Webdirect folder. The “`webdirect/lib`” and “`webdirect/xml`” folders contain the necessary files for a Jakarta container, while the JEE related files will be stored in the “`webdirect/lib/jee`” and “`webdirect/xml/jee`”.

As shown in Figure 1, IsCOBOL WebDirect installer, the developer can select the appropriate Jakarta or JEE servlet libraries depending on the container used for deploying.

**Figure 1.** isCOBOL WebDirect installer.

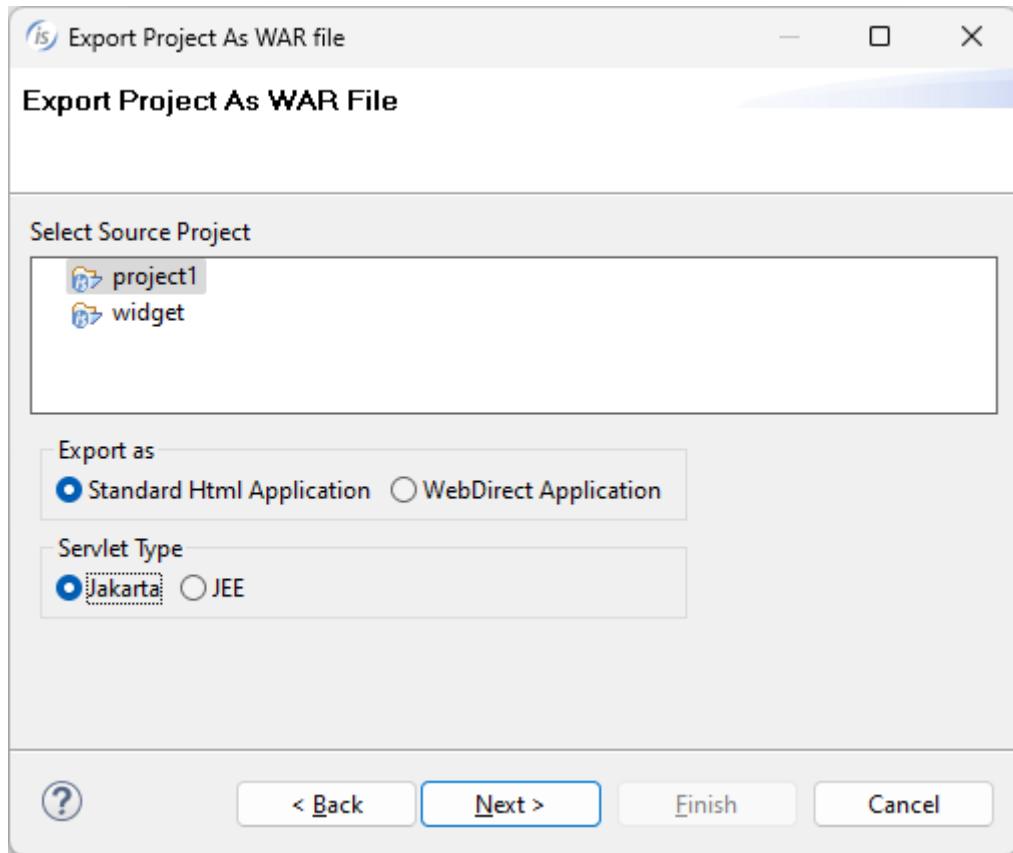


## IsCOBOL IDE

isCOBOL 2024 R1 IDE supports Jakarta, and the feature “Export Project as WAR file” that creates a .war file to be deployed in a Java container now asks the user to select the Servlet Type and will generate the correct libraries and configuration for the selected scenario.

This feature is available for isCOBOL projects that use WebDirect and for projects that use the isCOBOL HTTPHandler class, for example applications that leverage the WebServices bridge generation. As shown in Figure 2, Export to WAR file, the developer can select Jakarta or JEE in the Servlet Type area depending on the container used when deploying the application.

**Figure 2.** Export to WAR file.



## ESQL enhancements

isCOBOL 2024 R1 contains many enhancements in the ESQL area. Use SQLJ code to improve performances in batch elaborations with a new compiler option. Simplify the work to move a COBOL application from one database to another with new configurations and interfaces. In addition, the compatibility with DB2prep has been improved.

### SQLJ support

The isCOBOL compiler can now generate SQLJ code to be used instead of standard JDBC APIs on Oracle and DB2 databases. SQL is a non-procedural language for defining and manipulating data in relational databases. SQLJ is a language that embeds static SQL in Java in a way that is compatible with Java's design philosophy. SQLJ does syntax-checking of the embedded SQL, type-checking to assure that the data exchanged between Java and SQL have compatible types and proper type conversions, and schema-checking to assure congruence between SQL constructs and the database schema.

Using SQLJ instead of standard JDBC APIs on Oracle and DB2 databases provides better performance of the static SQL, including every DML query that is not prepared. To generate program classes that manage ESQL via SQLJ instead of standard JDBC APIs, add the -sqlj option to your compiler command line. For a correct result the "sqlj" command (the SQLJ translator) must be available in the Path environment. To see the difference between standard JDBC APIs and SQLJ, you can use -jj -jc in the compiler command line and look at the intermediate java source.

A query like this:

```
EXEC SQL
  INSERT INTO TBL (COD, NAME, ADDRESS)
    VALUES (:WK-COD, :WK-NAME, :WK-ADDR)
END-EXEC.
```

is translated to a call to the isCOBOL's ESQLRuntime based on JDBC APIs as:

```
ESQL_CURS_HNDL.set(Esql.DECLARE_STMT(new Object[] {
    SQLCA,null, null, "INSERT INTO TBL(COD, NAME, ADDRESS) VALUES ( ?, ?, ?)", null, "0", null, "0"}));
RETURN_CODE.set(Esql.SET_PARAM(new Object[] {
    ESQL_CURS_HNDL, WK_ADDR, ESQL_BIND_TYPE,Factory.getNumLiteral(3, 4, 0, false),null, null, $0$, "INOUT")));
RETURN_CODE.set(Esql.SET_PARAM(new Object[] {
    ESQL_CURS_HNDL, WK_NAME, ESQL_BIND_TYPE,Factory.getNumLiteral(2, 4, 0, false),null, null, $0$, "INOUT")));
RETURN_CODE.set(Esql.SET_PARAM(new Object[] {
    ESQL_CURS_HNDL, WK_COD, ESQL_BIND_TYPE,Factory.getNumLiteral(1, 4, 0, false),null, null, $0$, "INOUT")));
RETURN_CODE.set(Esql.EXECUTE(new Object[] {
    SQLCA, ESQL_CURS_HNDL}));
```

When the -sqlj switch is used the query is translated to this code:

```
#sql [ctxINS] {INSERT INTO TBL(COD, NAME, ADDRESS)
  VALUES ( :sqlj$p$WK_COD, :sqlj$p$WK_NAME, :sqlj$p$WK_ADDR)};
```

To compile and run the isCOBOL's ESQL sample on DB2 with SQLJ, follow these steps:

1. Compile the program with the -sqlj option, e.g.:

```
iscc -sqlj ESQL-SAMPLE.cbl
```

2. Bind the generated profile file to the database, e.g.

```
db2prof " jdbc:db2://localhost:50000/SAMPLE" -user db2inst1 -password secret
ESQL_SAMPLE_SJProfile0.ser
```

3. Run the program:

```
iscrun ESQL_SAMPLE
```

A table of performance gains is shown in Figure 3, Comparing JDBC vs SQLJ, where the same program is executed with both the standard JDBC and SQLJ. The test was run in 64-bit Ubuntu Linux on an Intel Core i5 Processor 4440+ clocked at 3.10 GHz with 16 GB of RAM, using Oracle JDK 1.8.0\_381 and IBM DB2 11.5. Times are expressed in seconds. The COBOL program reads 96,000 rows from a table using single SELECT statements:

```
EXEC SQL
  SELECT field-1, field-2, ..., field-n
    INTO :hostvar-1, :hostvar-2, ..., :hostvar-n
    FROM table-name WHERE id = ?
END-EXEC
```

The test has been repeated 3 times to get an average time.

**Figure 3.** Comparing JDBC vs SQLJ.

Iteration	N. queries	Time JDBC	Time SQLJ
1	96,000	28.50	17.91
2	96,000	29.43	18.50
3	96,000	28.66	19.10
<b>Average</b>	96,000	<b>28.86</b>	<b>18.50</b>

## Move to a different Database

When COBOL application written with ESQL code for a specific RDBMS need to be migrated to a different RDBMS, developers face the challenge of adjusting the SQL code that may have been written to support a specific RDBMS and is not compatible with another RDBMS. isCOBOL 2024 R1 can ease the migration with both the improved PreProcessor, which performs manipulation of Embedded SQL statements enclosed in the "EXEC-SQL" and "END-EXEC" statements, and the new configuration options and interfaces to customize handling of ESQL by the runtime framework.

The first step is to write a PreProcessor class using either Java or COBOL, and use it during the compilation to adjust the static ESQL statements used in the source. To use the PreProcessor it's necessary to add the PreProcessor class name in the compiler configuration file.

For example, the query:

```
EXEC SQL
  SELECT CHAR(CURRENT_DATE, ISO)
    INTO :wrk-date
    FROM SYSIBM.SYSDUMMY1
END-EXEC.
```

uses valid syntax in DB2 to retrieve the current date, but it is not supported by Oracle. The PreProcessor needs to change the code to:

```
EXEC SQL
  SELECT TO_CHAR(CURRENT_DATE, 'YYYY-MM-DD')
    INTO :wrk-date
    FROM DUAL
END-EXEC.
```

The new query can run on Oracle using its built-in TO\_CHAR function instead of DB2's CHAR, and the parameters passed to the function are different. Also the FROM clause needs to be changed to be supported by Oracle. The PreProcessor code needed to perform this change is provided as a sample in the isCOBOL installation.

The second step is to manage the same syntax replacement for dynamically generated queries, since those can't be identified in the source or are not present in the source code at all. A new configuration named `iscobol.esql.prepare_handler` has been implemented to allow developers to provide a class that customizes the PREPARED ESQL statements before they are executed. For example, the following code:

```
MOVE "SELECT CHAR(CURRENT_DATE, ISO) FROM SYSIBM.SYSDUMMY1"
  TO WRK-QUERY.
EXEC SQL
  PREPARE CMD FROM :wrk-query
END-EXEC.
```

executes the same query shown before but with the PREPARE statement. It will be intercepted by the class that implements the new interface, which can then perform the needed changes to be compatible to the target RDBMS. The class must implement the interface com.iscobil.rts.EsqlPrepareHandler, which requires the following method to be defined:

```
public void queryDecoder(CobolVar query)
```

The queryDecoder method allows you to alter the SQL query text of a prepared ESQL statement before it is executed and the same logic implemented in the previous PreProcessor class can be reused. To have your class be automatically called after each ESQL PREPARE, set *iscobil.esql.prepare\_handler=<classname>* in the configuration. The installed sample also demonstrates how to use this configuration in conjunction with the PreProcessor code.

Different RDBMs can return different status codes, so the last step involves the configuration of SQLCA fields to map the error codes to common codes. The new interface com.iscobil.rts.EsqlSqlcaHandler can ease this task by implementing the following method:

```
public void sqlcaDecoder(SQLException ex,
                         CobolVar sqlcode,
                         CobolVar sqlstate,
                         CobolVar sqlerrmc)
```

You can use this method to set the SQLCODE, SQLSTATE and SQLERRMC fields before they're returned to the COBOL program. The method receives as input the instance of java.sql.SQLException that was raised then running the query, and you can inquire to get the error details and set SQLCA fields accordingly.

To have your class automatically called after each ESQL error, set *iscobil.esql.sqlca\_handler=<classname>* in the configuration.

This feature integrates the existing *iscobil.esql.sqlcode.<value>=<new-value>* configuration setting.

With *iscobil.esql.sqlcode.<value>=<new-value>* you can map a SQLCODE value to another, creating a compatibility between different databases. But *iscobil.esql.sqlcode.<value>=<new-value>* cannot be used with databases like PostgreSQL, where the SQLCODE is 9999 for every error. In this case, you can set SQLCODE according to the SQLException with a custom class implementing the com.iscobil.rts.EsqlSqlcaHandler interface.

## DB2 compatibility

COBOL programs can now expose ESQL cursors to their callers, who can now access the cursor data through an array of java.sql.ResultSet objects. This feature is particularly useful for sharing sets of data with Java programs that call COBOL programs using the com.iscobil.java.IsCobol class, with or without the EasyLinkage facility.

isCOBOL now supports the "WITH RETURN" clause in the DECLARE CURSOR statement, and you can define a cursor as follows:

```
EXEC SQL
  DECLARE sharedcur CURSOR WITH RETURN
    FOR
      SELECT CLI_COD, CLI_NAME, CLI_ADDRESS
        FROM CLIENTS_TBL
  END-EXEC
```

The COBOL program should just open the cursor and leave it open, without doing any FETCH on it.

In the Java program, two new methods are available in the com.iscobol.java.IsCobol class:

- public void registerResultSets() : this method must be called before calling the COBOL programs and instructs the isCOBOL Framework to collect all the cursors that were declared with the WITH RETURN clause that have been opened and have not been closed.
- public ResultSet[] getResultSets() : this method can be called after calling the COBOL programs and returns the array of ResultSet objects that store cursors data. For every COBOL cursor collected by the Framework a java.sql.ResultSet is returned. The Java program can scan these ResultSet objects to get the data. When done, it can close these ResultSet to release allocated memory.

## Compiler enhancements

The Compiler supports new syntax to allow inline variable declaration and improved the auto-boxing in Object Oriented Programming. A new compiler option has been implemented to create an external DataMap that declares all data items of DATA DIVISION.

### Inline variable declaration

Many languages such, as C# and Java, support the inline variable declaration syntax that allows you to declare a local variable directly in a code block. This has different advantages, such as:

- no risk of creating conflicts by reusing the same variable in a different code block
- thread safe code related to the variable when running multi-threading programs
- fast code in loops since the inline variables use primitive types

The newest isCOBOL compiler implements primitive type inline variable declaration in the AS clause of PERFORM VARYING statements. For example, code like this:

```
perform varying in-idx as "int" from 1 by 1 until in-idx > 10000
    move my-var(in-idx) to back-var(in-idx)
    ...
end-perform
```

can be used to declare the in-idx variable, which is not declared in the DATA DIVISION but can be accessed in the code block between PERFORM VARYING and END-PERFORM.

It is basically the equivalent of, but performs better than, the following:

```
working-storage section.
01 w-idx      object reference "int".
...
    perform varying w-idx from 1 by 1 until w-idx > 10000
        move my-var(w-idx) to back-var(w-idx)
        ...
    end-perform
```

The inline variable type declaration follows the same syntax as the OBJECT REFERENCE clause and can reference a Java class name or a logical call name defined in the REPOSITORY.

## Improved auto-boxing

Auto-boxing refers to the capability of mixing Java and primitive types with COBOL variables in the same statement, making source code more readable and flexible. Auto-boxing has been vastly improved in the 2024 R1 release by supporting:

- the comparisons between numeric Java data types and COBOL. For example, code such as:

```
working-storage section.  
77 obj-int      object reference "int".  
77 obj-long     object reference "long".  
77 obj-lang-int object reference "java.lang.Integer".  
77 obj-lang-long object reference "java.lang.Long".  
77 var-num      pic 9(5)v9(3).  
...  
    if obj-int > 10  
    if obj-int = var-num  
    if var-num < obj-lang-long  
    perform varying var-num from 1 by 1 until var-num > obj-long  
    perform parag2 until obj-lang-int = var-num  
...
```

can now be compiled and executed, making it simpler to compare a numeric COBOL data item with other Java data types and primitive types.

- COBOL arithmetic operations on primitive data types and objects. The COBOL statements ADD, SUBTRACT, MULTIPLY, DIVIDE and COMPUTE can now contain Java data types without limitations. For example, code such as:

```
add 1 to obj-int  
subtract var-num from obj-long  
multiply obj-lang-int by var-num giving obj-lang-long  
divide obj-lang-long by obj-lang-int giving var-num  
compute obj-lang-long = (var-num * 2) + 1
```

can be used to mix COBOL variables and Java data types in the same statement.

- arithmetic operations in CALL and INVOKE parameters. This is typically useful when passing parameters BY VALUE in CALL and when passing numeric parameters to Java methods. For example, the following code is now allowed:

```
call "PROG" using by value var-num - 1  
                  by value var-num + 1  
if obj-string:>substring(var-num + 1, var-num + 2) = "A"
```

## External DataMap

The new compiler option -edm can create an xml file with the application's DataMap. This file contains all the fields declared in DATA DIVISION; describing the details, declaration location and usage information. This xml file can be used to perform analysis with external tools or can be imported in a RDBMS system and queried. It can also be used to generate documentation by developers, for example.

Using the compiler option -edo=<path> you can specify the location for the DataMap output file.

For example, compiling with this command:

```
iscc -edm -edo=datamap progcust.cbl
```

a source that contains:

```
working-storage section.  
 77 V1          pic 99.  
 01 G1-VARS.  
    03 G1-V1      pic 99.  
    03 G1-V2      pic x(10).  
linkage section.  
copy "def-params.def".
```

with the copyfile def-params.def containing:

```
01 DEF-PAR1      pic x(10).  
01 DEF-PAR2.  
  03 DEF-PAR-V3  pic 9(5).
```

This XML file can be used by an external tool to inquire all the information relative to every data-item; such as name, offset, length, type and declaration location (FD, Working, Linkage, ...). Additional details include whether the field is elementary, boolean, constant, occurs, redefines, external, used by the program, or used as parameter in the USING clause of CALL and INVOKE statements. You can also determine if the data-item is declared in the main source or in a COPY file and also the replacing clause in case of a COPY REPLACING.

The file progcust.xml is created in the datamap folder with the following content:

```
<program name="PROGCUST">  
  <field>  
    <name>V1</name>  
    <location>WS</location>  
    <offset>0</offset>  
    <physicalLength>2</physicalLength>  
    <dataType>NumUnsigned</dataType>  
    <elementary>yes</elementary>  
    <usedByProgram>yes</usedByProgram>  
    <usedAsParameter>yes</usedAsParameter>  
  </field>  
  <field>  
    <name>G1-VARS</name>  
    <location>WS</location>  
    <offset>0</offset>  
    <physicalLength>14</physicalLength>  
    <dataType>Alphanum</dataType>  
    <group>yes</group>  
    <usedByProgram>no</usedByProgram>  
    <usedAsParameter>no</usedAsParameter>  
  </field>
```

```

<field>
  <name>G1-V1</name>
  <location>WS</location>
  <offset>0</offset>
  <physicalLength>2</physicalLength>
  <dataType>NumUnsigned</dataType>
  <elementary>yes</elementary>
  <usedByProgram>no</usedByProgram>
  <usedAsParameter>no</usedAsParameter>
</field>
<field>
  <name>G1-V2</name>
  <location>WS</location>
  <offset>2</offset>
  <physicalLength>10</physicalLength>
  <dataType>Alphanum</dataType>
  <elementary>yes</elementary>
  <usedByProgram>no</usedByProgram>
  <usedAsParameter>no</usedAsParameter>
</field>
<field>
  <name>DEF-PAR1</name>
  <location>LS</location>
  <offset>0</offset>
  <physicalLength>10</physicalLength>
  <dataType>Alphanum</dataType>
  <elementary>yes</elementary>
  <usedByProgram>yes</usedByProgram>
  <usedAsParameter>no</usedAsParameter>
  <copyFile>p2-params.def</copyFile>
</field>
...
</program>

```

## Compatibility enhancements

isCOBOL 2024 R1 improves compatibility with IBM COBOL by supporting LINE LIMIT in the REPORT SECTION and by implementing additional functions.

### Report Section

The Report Section is a COBOL section supported by IBM that allows definition of a report. It produces a text file with the statements “initiate”, “generate” and “terminate”.

isCOBOL 2024R1 adds support for the LINE LIMIT clause to specify the maximum number of characters that can be written to a line. Characters exceeding the line limit are truncated from the output.

For example, this code:

```
report section.  
  rd my-report  
    control are final  
    page limit 22 lines  
    line limit 50  
    heading 1.  
  01 detail-line type de line plus 2.  
    02 line plus 1.  
      03 column 4 pic x(10) source field1.  
      03 column 20 pic x(20) source field2.  
      ...  
  open output F1.  
  initiate my-report.  
  ...  
  generate detail-line.  
  terminate my-report.  
  close F1.
```

sets the LINE LIMIT to 50. The runtime will truncate the output generated by the running program to the specified size.

## New functions

NUMVAL and NUMVAL-C are existing functions meant to convert an alphanumeric variable to a numeric variable, and NUMVAL-C can be used on values containing currency symbols and/or commas.

To improve compatibility with IBM COBOL, the new NUMVAL-F function has been implemented in isCOBOL 2024 R1. This function extends conversion support to number strings that contain an exponent value.

Every function can be tested with the specific function named TEST-NUMVAL\* that returns 0 if the argument passed conforms to the argument rules for the corresponding NUMVAL\* function, or the position of the first character encountered that invalidates the string.

A code snippet like this:

```
if function test-numval(varx) = 0  
  move function numval(varx) to varn  
end-if  
...  
if function test-numval-c(varx) = 0  
  move function numval-c(varx) to varn  
end-if  
...  
if function test-numval-f(varx) = 0  
  move function numval-f(varx) to varn  
end-if
```

is now valid and can be compiled and executed successfully.

## Runtime enhancements

The runtime 2024 R1 has been enhanced with new configuration options and a new library routine.

## New configurations

When a COBOL application employs thread programming, leveraging statements like CALL THREAD or PERFORM THREAD, it becomes challenging to track the program's execution flow due to multiple threads running concurrently. While it is feasible to navigate through threads step by step in a debugger, there are instances when it is crucial to observe the simultaneous execution of all threads. This reflects the actual scenario when the program is running in a live production environment.

Analyzing runtime execution has been simplified with the introduction of a new configuration, namely `iscobol.logfile.thread=true`. This configuration enables the tracking of thread numbers at the end of each log line.

For example, a code such as:

```
perform thread DO-CHECK
perform thread DO-CHECK
...
DO-CHECK.
call "C$SLEEP" using 1
set environment "myevn" to varx
...
```

when running using the following configuration settings:

```
iscobol.logfile=isc.log
iscobol.tracelevel=1039
```

produces the following content in the log file:

```
...
12... INFO: ENTER PARAGRAPH 'DO-CHECK' [PROGA]
12... INFO: ENTER PARAGRAPH 'DO-CHECK' [PROGA]
12... INFO: ENTER isCOBOL LIB 'C$SLEEP' {
12... INFO: ENTER isCOBOL LIB 'C$SLEEP' {
12... INFO: SET ENVIRONMENT [users] 'iscobol.myevn=cust2'
12... INFO: SET ENVIRONMENT [users] 'iscobol.myevn=cust1'
...
```

It's possible to see that the paragraph DO-CHECK has been executed 2 times, but it's unclear if it has been executed by the same thread or two different threads.

Adding the new configuration:

```
iscobol.logfile.thread=true
```

the output becomes:

```
...
12... INFO: ENTER PARAGRAPH 'DO-CHECK' [PROGA] [PROGA Thread=4]
12... INFO: ENTER PARAGRAPH 'DO-CHECK' [PROGA] [PROGA Thread=5]
12... INFO: ENTER isCOBOL LIB 'C$SLEEP' { [com.iscobol.lib.C$SLEEP Thread=4]
12... INFO: ENTER isCOBOL LIB 'C$SLEEP' { [com.iscobol.lib.C$SLEEP Thread=5]
12... INFO: SET ENVIRONMENT [users] 'iscobol.myevn=cust2' [PROGA Thread=5]
12... INFO: SET ENVIRONMENT [users] 'iscobol.myevn=cust1' [PROGA Thread=4]
...
```

The log file now contains the thread ID that executed a statement, making it easier to follow the program flow.

A new configuration, iscobel.terminal.paste\_key, allows users to customize the key used for pasting in character-based accept statements. By default, pasting is accomplished through a mouse right-click. With the new configuration, it is now possible to modify the key combination or add additional keys for the same action. For instance, by setting:

```
iscobel.terminal.paste_key=mrdw,*p
```

Ctrl+P can be used for pasting in addition to the mouse right click.

### New routine

A new routine named C\$PARAMNAME has been implemented to retrieve the name of the parameter passed from the caller program. This information can be useful if specific code needs to be executed based on the parameter.

The called program, if needed, could use the output generated by the new external DataMap using the -edm compiler option and apply custom logic based on the name, level, or structure of the parameter received by the calling program.

For example, if a caller program executes this code

```
77 MY-VAR      PIC X(20) .
01 MY-GROUP .
 03 MY-VAR-1    PIC X(10) .
 03 MY-VAR-2    PIC 9(6)V99.
 03 MY-VAR-3    PIC X(5) .
...
CALL "PRCUST" USING MY-VAR, MY-GROUP
```

In the called program, the following code:

```
77 paramName  pic x(30) .
...
move 1 to paramNum
CALL "C$PARAMNAME" USING paramNum, paramName
          GIVING ret-code
...
add 1 to paramNum
CALL "C$PARAMNAME" USING paramNum, paramName
          GIVING ret-code
```

can be used to retrieve the string "MY-VAR" and "MY-GROUP", the parameter names specified in the caller program, in the paramName variable. The ret-code variable specified in the giving clause will contain 1 if the paramName can be set correctly, and will contain 0 if the specified parameter is a constant or was omitted.

## GUI enhancements

IsCOBOL Evolve 2024 R1 GUI grid capabilities have been enhanced by implementing new properties to customize colors. A new property input-filter is now supported to better filter the input.

## Grid colors

The grid control has been improved and now you can set the rollover row color. Setting the row-rollover-color or alternatively row-rollover-foreground-color and row-rollover-background-color properties in the grid control will make the entire row be painted with the specified colors when the mouse is over a cell.

When the mouse hovers in a header cell, the single cell is painted with the color properties specified in the new heading-rollover-color or heading-rollover-background-color and heading-rollover-foreground-color.

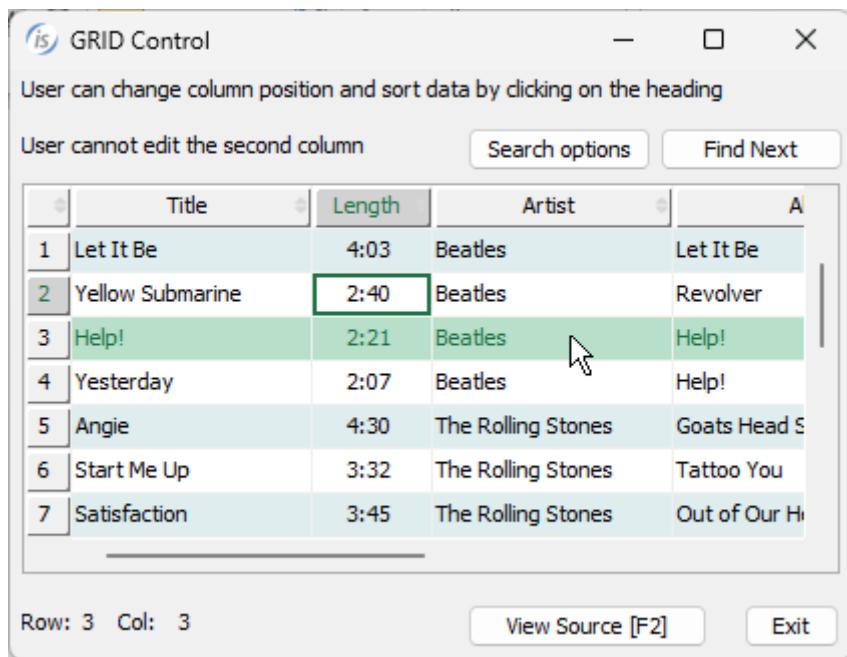
The new cursor-frame-color can be used to apply a specific color to the borders of a cell that has focus.

For example, this is the code where the colors of the grid are set:

```
03 Gd grid boxed vscroll
  column-headings row-headings centered-headings tiled-headings
  adjustable-columns reordering-columns sortable-columns
  row-background-color-pattern      (-16777215, -14675438)
  end-color                         -16774581
  heading-color                     257
  border-color                       rgb x#ACACAC
  heading-cursor-background-color   rgb x#D2D2D2
  heading-cursor-foreground-color   rgb x#217346
  cursor-frame-color                rgb x#217346
  row-rollover-background-color     rgb x#B7DFC9
  row-rollover-foreground-color     rgb x#217346
  heading-rollover-background-color rgb x#9FD5B7
  heading-rollover-foreground-color rgb x#000000
  ...
  
```

The result of the program running is shown in Figure 4, Grid colors. The row covered by the mouse pointer is now more noticeable, and the cursor frame color has been harmonized to better blend with the colors used in the headings, providing clearer indications of the cursor's position.

**Figure 4.** Grid colors.



## Input-Filter property

A new property named input-filter is now supported in controls that accept data input, such as the Entry-Field, Combo-Box and Grid, giving the developer the ability to better filter the text users can type. A regular expression is used in this property, making it extremely flexible.

A code snippet like this:

```

03 ef1 entry-field
  line 3 col 2 size 67 cells
  input-filter "[A-Za-z]+"
  ...
03 cb1 combo-box drop-down unsorted
  line 6 col 2 size 67 cells
  input-filter "[A-Z\sa-z]+"
  ...
03 Gd grid
  line 10 col 2 lines 9 size 67 cells
  display-columns (1, 5, 25, 35, 55, 80, 100, 120)
  data-columns (1, 4, 34, 39, 59, 89, 104, 134)
  input-filter ("*", "*", "[0-9:]+", "*",
               "*", "*", "*", "[0-9] +")
  ...

```

defines different rules of acceptable input data in the controls:

- the entry-field accepts letters without spaces
- the combo-box accepts letters and spaces
- the grid accepts numbers and ":" in the third column and only numbers in the last column. The rest of the columns have no restrictions.

# GIFE enhancements

The GIFE (Graphical Indexed and relative File Editor) utility, has been improved in the 2024R1 release to add a new view that lets you see multiple records in a grid where every line is a record and every column is a field of the file. It also now supports files with 01 level redefines or conditional 01 levels in the FD declarations.

## New List view

When an indexed or relative file along with the EFD file created by the compiler option -efd is opened by GIFE, a new view named List view is available to see multiple records in a grid. This view is read-only, so to change the content or insert a new record the previous Field view should be used.

As shown in Figure 5, New List view, the records are loaded in grid rows. Using the buttons on the bottom pane, additional records can be loaded and viewed.

**Figure 5.** New List view.

#	AN_COD	AN_NAME	AN_ZIP	AN_DATE	AN_AMOUNT
1	1	name 000000001	12345	20230319	1232500.000000
2	2	name 000000002	619	20070203	1358.450000
3	3	name 000000003	0	20051231	2500.000000
4	4	name 000000004	0	20051231	2500.000000
5	5	name 000000005	0	20051231	2500.000000
6	6	name 000000006	0	20051231	2500.000000
7	7	name 000000007	0	20051231	2500.000000
8	8	name 000000008	0	20051231	2500.000000
9	9	name 000000009	0	20051231	2500.000000
10	10	name 000000010	0	20051231	2500.000000
11	11	name 000000011	0	20051231	2500.000000
12	12	name 000000012	0	20051231	2500.000000
13	13	name 000000013	0	20051231	2500.000000
14	14	name 000000014	0	20051231	2500.000000
15	15	name 000000015	0	20051231	2500.000000
16	16	name 000000016	0	20051231	2500.000000
17	17	name 000000017	0	20051231	2500.000000

The search box can be used to filter the data and load only the records that meet the filter criteria.

## Condition support

Veryant products such as Database Bridge and UDBC driver fully support files with multiple or conditional 01 levels in the field declaration. This information is stored in an .xml file generated using the -efd compiler option, and now GIFE can use this information to better display the records of such files in the Field view.

The following FD definition contains multiple 01 level definition, based on conditions:

```
fd  filem.  
$EFD WHEN F_TYPE = "M" TABLENAME = AMERICAN_PEOPLE  
01  f-recM.  
03  f-key.  
    05 f-cod          pic 9(3).  
    05 f-type         pic x.  
03 american-person.  
    05 a-first-name   pic x(32).  
    05 a-second-name  pic x(32).  
    05 a-address      pic x(32).  
    05 a-zip          pic x(5).  
    05 add-field-1    pic x(10).  
    05 add-field-2    pic x(10).  
$EFD WHEN F_TYPE = "E" TABLENAME = EUROPEAN_PEOPLE  
01  f-recE.  
03  f-keyE.  
    05 f-codE         pic 9(3).  
    05 f-typeE        pic x.  
03 european-person.  
    05 E-first-name   pic x(32).  
    05 E-second-name  pic x(32).  
    05 E-address      pic x(32).  
    05 E-zip          pic x(5).  
    05 new-field-A    pic x(5).  
    05 filler         pic x(15).  
...
```

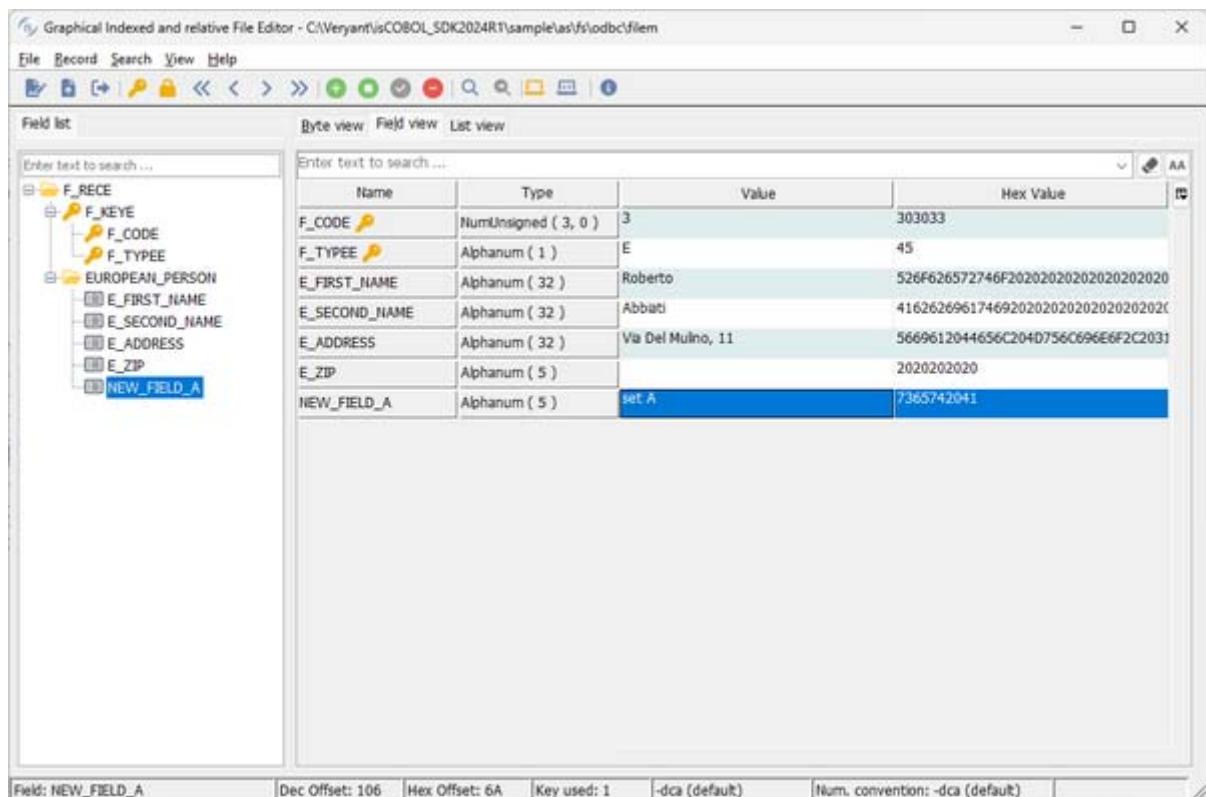
and is now fully supported by GIFE.

As shown in Figure 7, Condition for American record, and Figure 8, Condition for European record, the fields loaded in the Field view are the ones listed in the specific 01 level.

When reading different records, depending on which condition is satisfied, the relevant fields are displayed in the Field view.

**Figure 7.** Condition for American record.

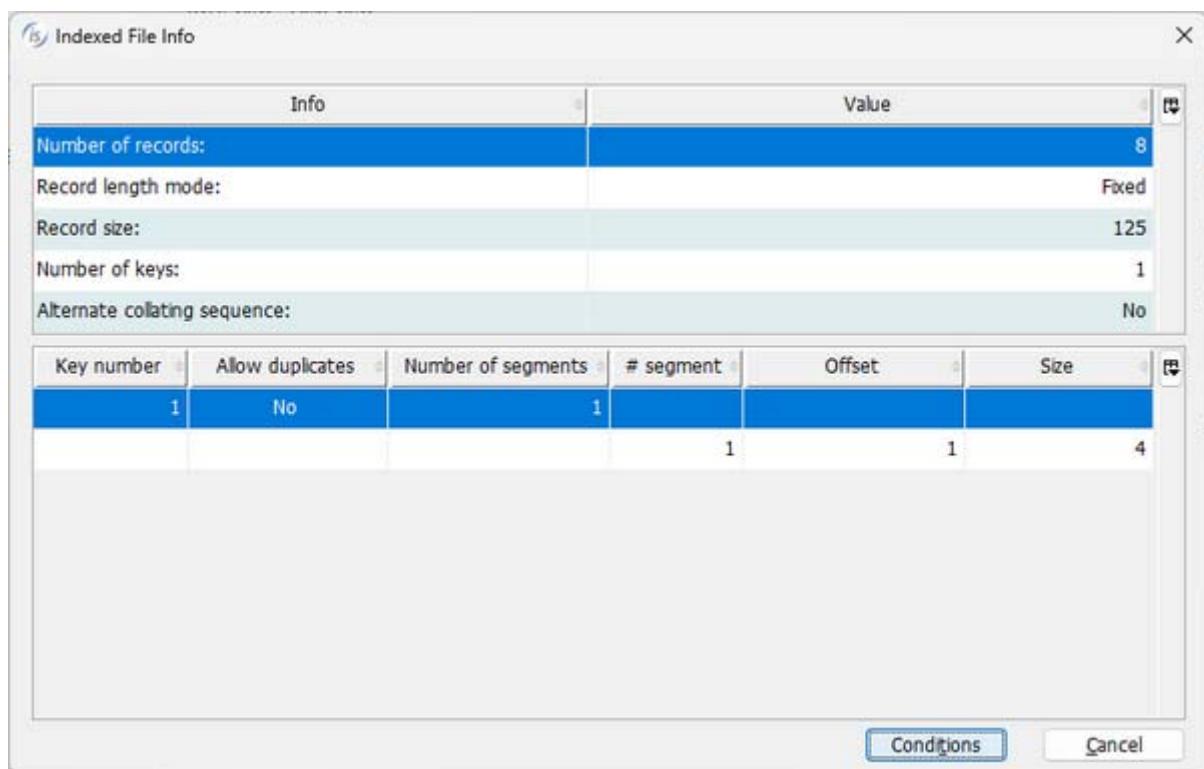




It's also possible to list all the conditions present in the file by opening the File Info and then pressing the button named "Conditions". This opens an additional window as shown in Figure 9, File Info, and Figure 10, Conditions.

The same window with the list of conditions is also opened when inserting a new record that allows you to pick which 01 level to use when setting the fields contents.

**Figure 9.** File Info.



**Figure 10.** Conditions.

#	Field	Table	Condition expression
01	F_REC1	AMERICAN_PEOPLE	F_TYPE = "M"
02	F_REC2	EUROPEAN_PEOPLE	F_TYPE = "E"
03	F_REC3	ASIAN_PEOPLE	F_TYPE = "S"
04	F_REC4	AFRICAN_PEOPLE	F_TYPE = "F"
05	F_REC5	AUSTRALIAN_PEOPLE	F_TYPE = "U"

Buttons: Select, Cancel.

# isCOBOL Evolve

---

## isCOBOL 2023 Release 1 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2023 R1.

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

Starting from this release, an extension for Visual Studio Code is provided to edit, compile and debug isCOBOL sources and projects.

The new c-treeRTG release supports the web tool Data Replication Manager to easily configure and monitor replications from a web browser.

The new WebClient release includes the new Test Tool product that provides test automation.

isCOBOL Evolve 2023 R1 supports new compatibility features to simplify the migration of existing COBOL programs to the isCOBOL Evolve suite.

Details on these enhancements and updates are included below.

### Visual Studio Code

Visual Studio Code, also known as VS Code, is a source-code editor made by Microsoft for Windows, Linux and macOS. It is very popular and customizable using extensions. Veryant isCOBOL Extension for Visual Studio Code allows COBOL developers to create and manage isCOBOL projects, edit sources, and run and debug isCOBOL programs. Settings are available to configure the location of the isCOBOL SDK, the compiler options used when compiling, the run options, the editor and much more.

These settings can be global, which are valid for any project, or project-specific.

The Veryant isCOBOL Extension provides several useful new commands, accessible using the shortcut key Shift-Ctrl-P on Windows and Linux and Shift-Cmd-P on MacOS. COBOL developers can use these commands to create new isCOBOL projects, create new source and copybook files with a default template and useful source code helper functions, compile source code, and run debugger-specific commands.

When compiled in debug mode, the VS code debugger can be used to debug isCOBOL programs, supporting common debugging features such as stepping in code, setting breakpoints, evaluating variables and setting watches.

Debugging is supported starting from isCOBOL 2023 R1, while editing is supported for all isCOBOL versions.

As shown Figure 1, *Editing in the Veryant isCOBOL Extension*, the COBOL source is edited and compiled in VS Code. Figure 2, *Debugging in the Veryant isCOBOL Extension*, shows the isCOBOL Debugger is in action.

Figure 1. Editing in the Veryant isCOBOL Extension.

The screenshot shows the Visual Studio Code interface with the "isCONTROLSET.cbl" file open in the editor. The code is a COBOL program with several sections: source, copy, common.wrk, iscontrolsetsor, lookup-link.wrk, notifprog-link.wrk, object.wrk, pdf-attributes.wrk, printgui-link.wrk, songs.wrk, list, output, and source. The "source" section contains the main logic. The "copy" section includes code to copy various files like "object.wrk", "lookup-link.wrk", "notifprog-link.wrk", and "printgui-link.wrk". The "common.wrk" section contains a "screen section" with a "copy" statement for "iscontrolset.scr". The "iscontrolsetsor" section starts with a "procedure division" and "declaratives". The "lookup-link.wrk" section contains a "SET-GRID section" with a "use after standard-error procedure on data-gui". The "notifprog-link.wrk" section has a "continue" statement. The "object.wrk" section ends with "end declaratives". The "pdf-attributes.wrk" section begins with a "MAIN" section containing a "call" statement to "C\$SYSINFO". The "printgui-link.wrk" section contains "accept" statements for "system-information" and "terminal-abilities". The "songs.wrk" section ends with a "call" to "C\$GETRUNENV". The "list" section contains a "call" to "C\$GETRUNENV". The "output" section contains a "J ISCONTROLSET.class" entry. The "source" section contains the "ISCONTROLSET.cbl" file itself. The bottom status bar shows "compilation terminated successfully".

```
source > ISCONTROLSET.cbl
26   copy "object.wrk".
29   copy "lookup-link.wrk".
30   copy "notifprog-link.wrk".
31   copy "printgui-link.wrk".
32
33 screen section.
34   copy "iscontrolset.scr".
35
36 procedure division.
37   declaratives.
38
39 SET-GRID section.
40   use after standard-error procedure on data-gui.
41   continue.
42
43 end declaratives.
44
45 MAIN.
46   call client "C$SYSINFO" using system-information
47   if os-is-mac
48     set client-is-mac to true
49   end-if.
50
51 accept system-information from system-information
52
53 accept terminal-abilities from terminal-info.
54
55 call "C$GETRUNENV" giving env-code
```

Figure 2. Debugging in the Veryant isCOBOL Extension.

```
copy "lookup-link.wrk".
copy "notifprog-link.wrk".
copy "printgui-link.wrk".

screen section.
copy "iscontrolset.scr".

procedure division.
declaratives.
SET-GRID section.
use after standard error procedure on data-gui.
continue.
end declaratives.
MAIN.
call client "C$SYSINFO" using system-information
if os-is-mac
  set client-is-mac to true
end-if.

accept system-information from system-info.

call "C$GETRUNENV" giving env-code

set environment "gui.justify_num_fields" to "1"
set environment "gui.native_style" to "1"
set environment "gui.curr_bcolor" to 28-gui-curr-bcolor
set environment "test_cure_header_color" to 28-mid-cure-header-color
```

## WebClient Test Tool

The isCOBOL WebClient installer contains a new product named Test Tool. With Test Tool you can interactively create test cases for your isCOBOL character or graphical application running in WebClient. Test Tool is a web application that lets you record a test case, consisting of a series of mouse clicks and keyboard events a user performs when running an isCOBOL application, then set assertions on expected results and later playing back the events and checking that the application still works as intended. Multiple tests can be automated using Selenium Grid.

As shown in Figure 3, *WebClient Test Tool installer*, the new product has been selected to be installed along with the WebClient Server and Admin.

Figure 3. WebClient Test Tool installer



WebClient Test Tool is a separate service that can be installed on the same server where WebClient is running or on a different server. To start the service in foreground mode, issue the command:

```
webclient-testtool
```

To run the Windows service or Linux daemon in the background, the command is:

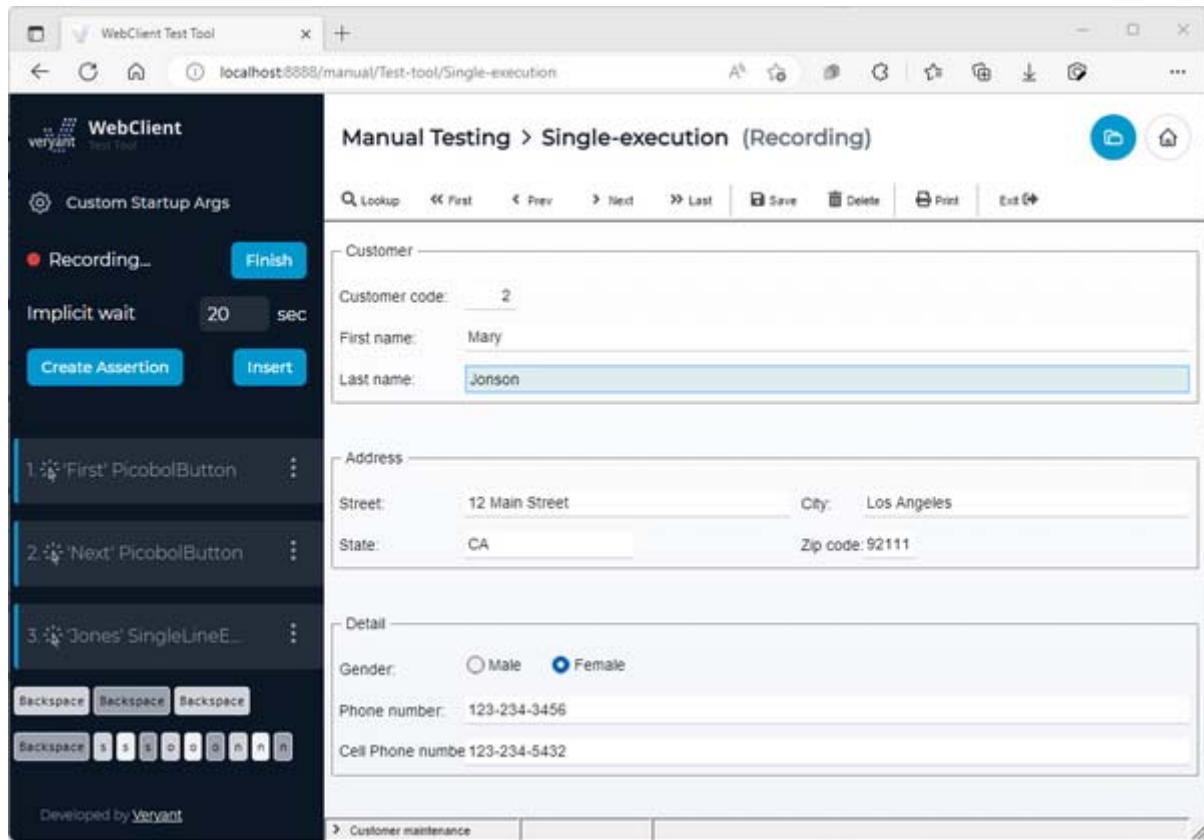
```
webclient-testtool -start
```

The Test Tool application can be reached using a web browser and navigating to the URL <http://localhost:8888/>.

From the html interface you can create test projects and start recording the actions taken when using the isCOBOL program running in WebClient. Every time a mouse click in the application is detected a new assertion is created. An assertion is a set of properties that will be validated in the test case, like component type, value, path, etc.

As shown in Figure 4, *WebClient Test Tool recording*, the right panel shows the WebClient application running and the left panel shows the list of recorded actions.

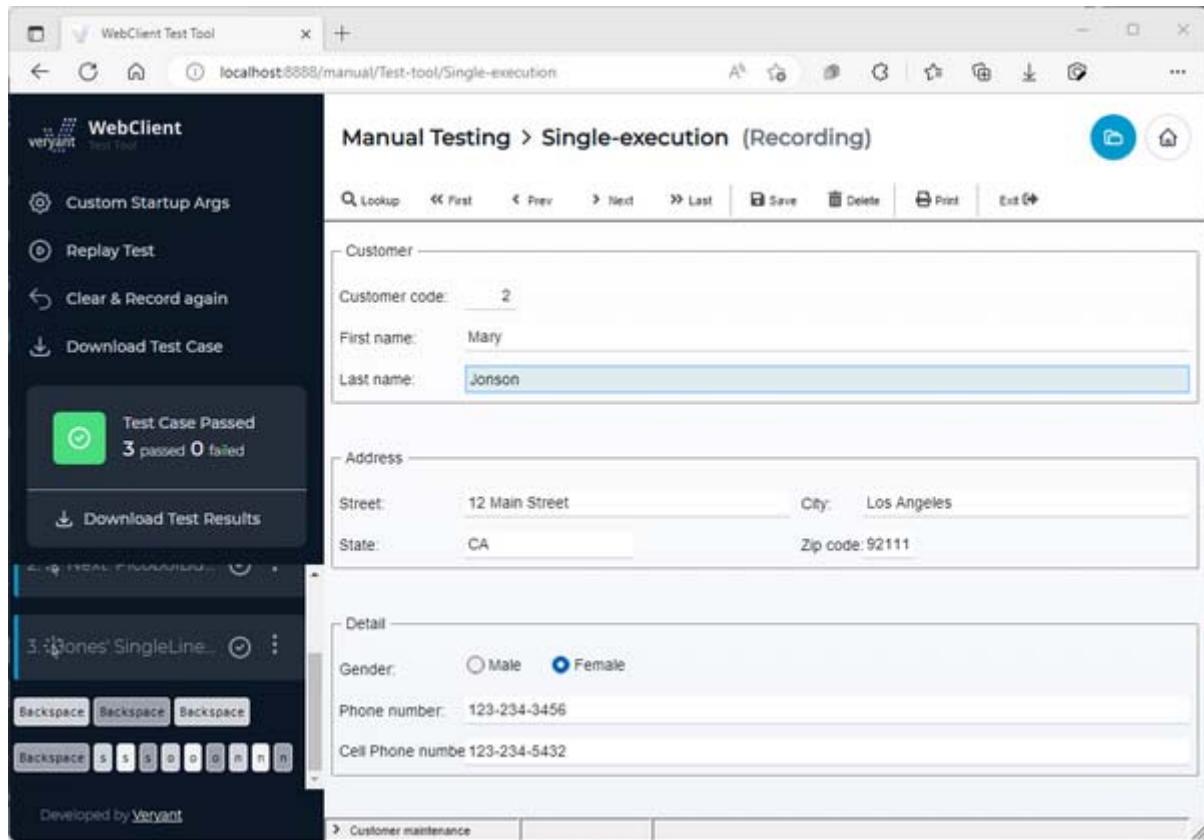
Figure 4. WebClient Test Tool recording.



When all the needed Assertion steps are created, click the Finish button, and the recorded actions are stored in a file in the project folder. The saved tests can be downloaded or played back to verify that changes in the isCOBOL application work as intended. Assertions are validated in the toolbox. Playback can be stopped, paused, and run step by step. Assertions can be edited or added, and breakpoints can be set just like when using a debugger.

Figure 5, *WebClient Test Tool result*, shows the isCOBOL program running in WebClient, and the results of the performed test cases.

Figure 5. WebClient Test Tool result.



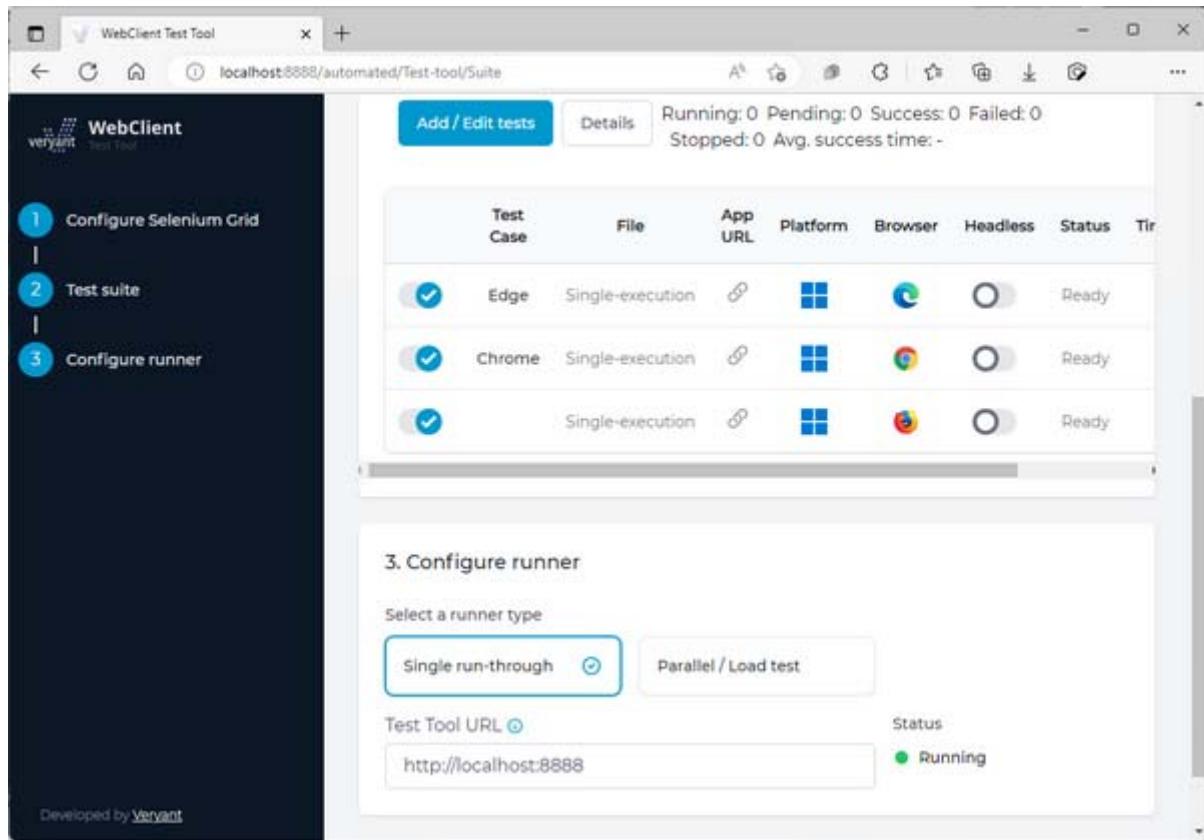
A test suite is a collection of tests that can be grouped to automate test cases.

A suite is a configuration file where you define which test cases to run and set options to be used. Automated test cases require Selenium Grid, a smart proxy server that makes it easy to run tests in parallel on multiple machines. This is done by routing commands to remote web browser instances, where one server acts as the hub. This hub routes test commands that are in JSON format to multiple registered Grid nodes.

A Selenium Grid can be set up either in your local environment or as a third-party service. To set the Selenium Grid in the WebClient Test Tool, set the URL of the running Selenium Hub, for example: <http://localhost:4444>. After the connection is validated, choose the platform and browser based on the availability of the environment on Selenium nodes. Finally, you can choose a “single run-through” or a “parallel run”. The first option executes test cases one-by-one, one instance at a time. The second option can run tests in parallel, and you can set the number of instances you want to run at the same time or after a timeout set in seconds.

As shown in Figure 6, *WebClient Test Tool Suite with Selenium*, the suite is ready to be executed on different browsers.

Figure 6. WebClient Test Tool Suite with Selenium.



When Selenium is used with WebClient Test Tool, it also provides a simple REST API to execute a test suite with one or more tests from an external application. A POST request can be sent to <http://localhost:8888/rest/runTest>, setting the Content-Type header to "application/json" with the following body:

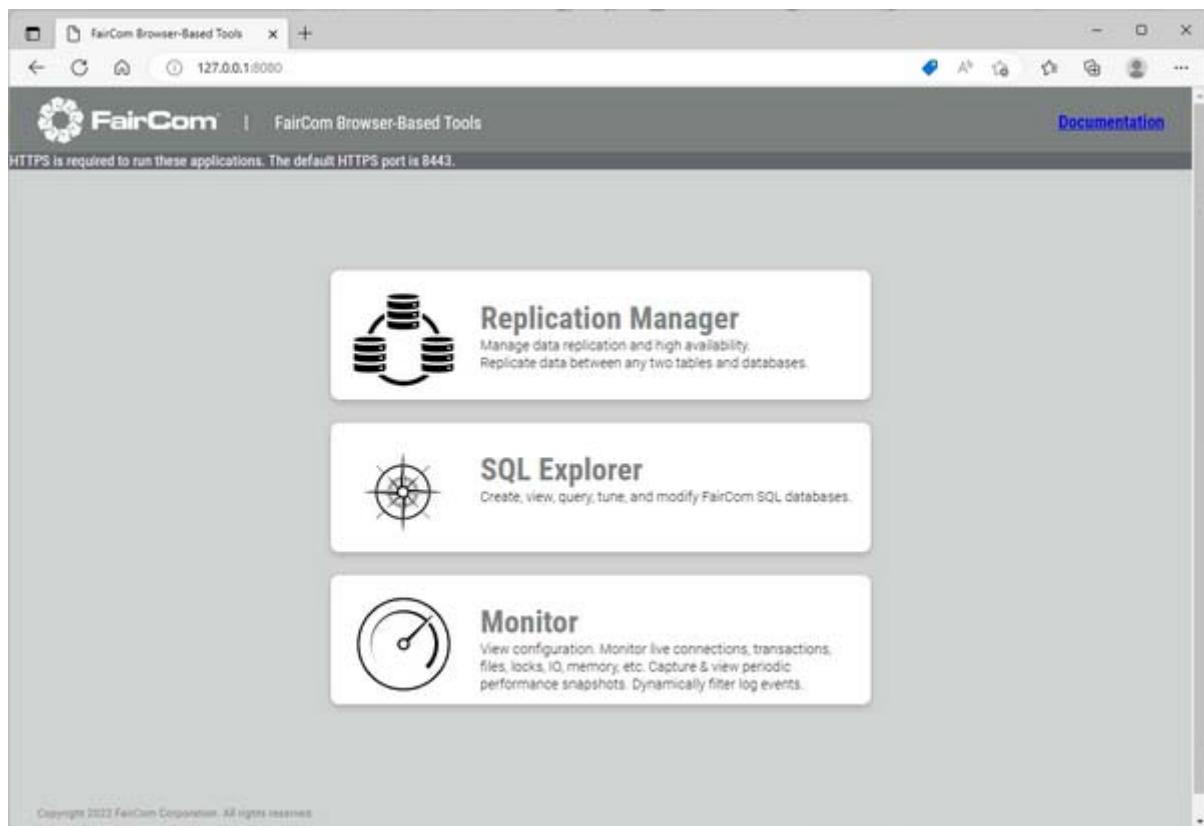
```
{
  "parallel": false,
  "maxRunningParallel": 1,
  "rampUpPeriod": 1,
  "runCount": 1,
  "testCaseParameters": [
    {
      "project": "project 1",
      "file": "test 1",
      "name": "Test",
      "webswingAppUrl": "http://localhost:8080/isapplication",
      "webswingUsername": "admin",
      "webswingPassword": "admin",
      "platform": "WINDOWS",
      "browser": "chrome",
      "headless": false,
      "enabled": true
    }
  ]
}
```

## C-Tree Replication Manager

The current v3.0.2 release supports a new product named Replication Manager. It is a web GUI tool you can use to easily configure replication between different c-tree servers. It can also be used to monitor the replication status. The Replication Manager, also known as Memphis, can be installed on the same server as c-tree server or on a different server. Memphis automatically detects the c-tree servers running that are configured to be managed by the replication manager.

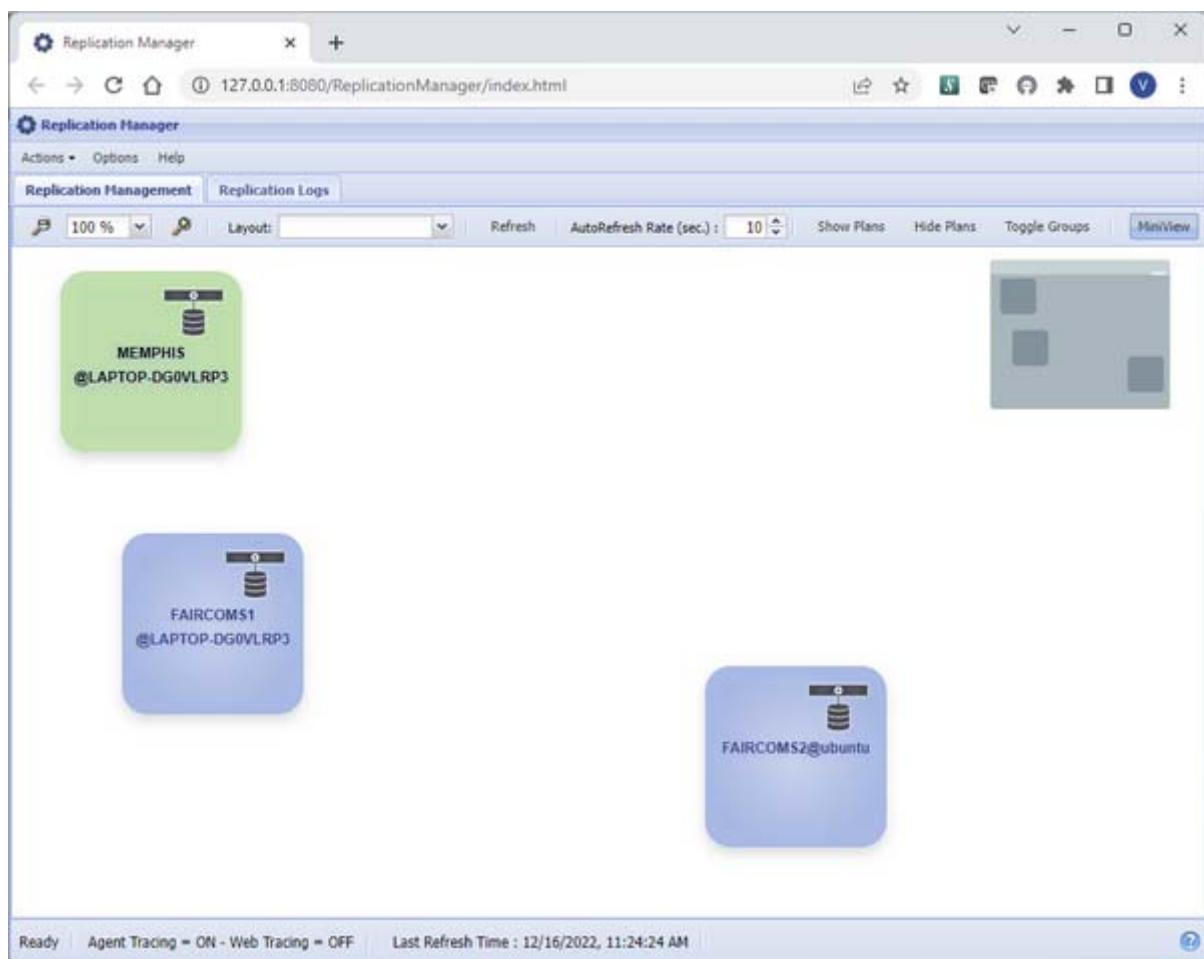
After starting the executable file named “Memphis” and the c-treeRTG servers in the network, you can open a browser and enter the URL: <http://127.0.0.1:8080> or <https://127.0.0.1:8443> to use the HTTPS protocol and see the main menu as shown in Figure 7, *Memphis web menu*,

Figure 7. Memphis web menu.



Clicking on the Replication Manager button opens a new browser tab with the Replication Manager application at URL <http://127.0.0.1:8080/ReplicationManager/index.html> and the list of available c-tree servers is shown as depicted in Figure 8, *Replication Manager View*.

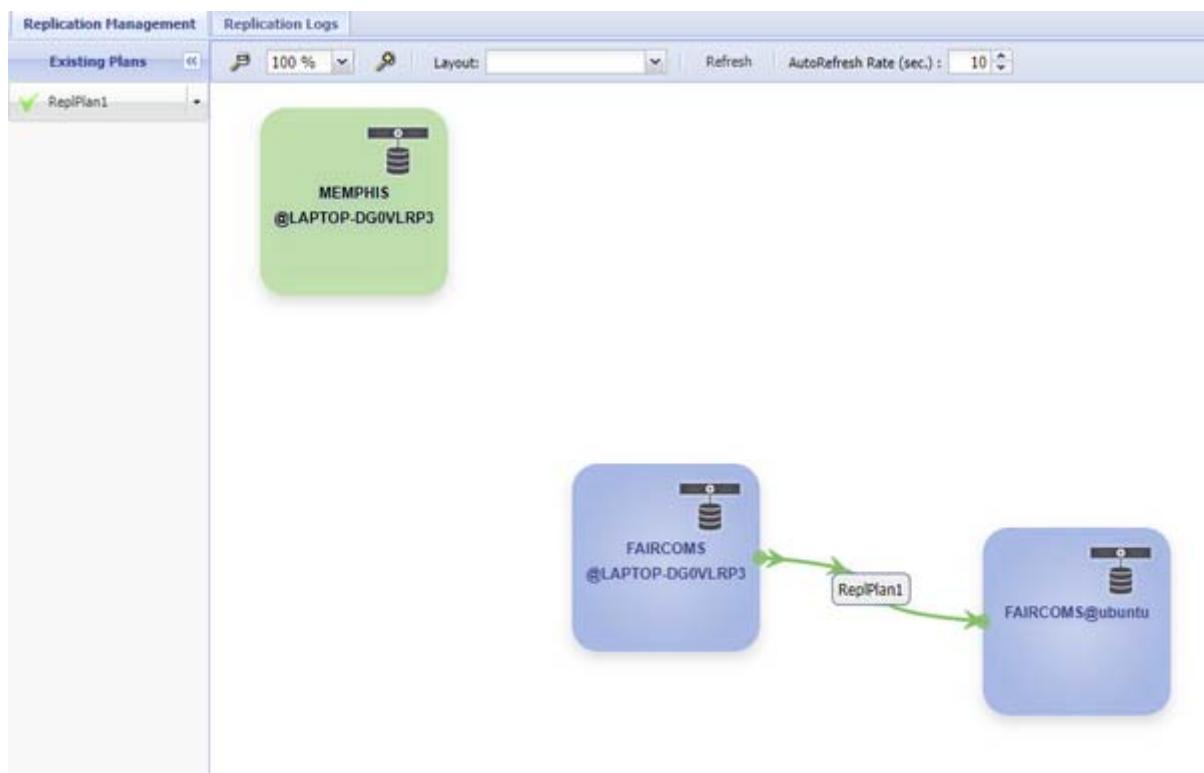
Figure 8. Replication Manager View.



The other links shown in Figure 7, *Memphis web menu* above can be used to open the SQL Explorer and Monitor web applications. These are the same as the existing c-tree web-tools and are now integrated in Memphis.

In the Replication Manager's application, you can drag a "Source" server over a "Target" server to create a Replication plan, based on a Publication where you can define if the replication is for SQL tables, ISAM files or both. You can also filter ISAM files included in specific folders that need to be included or excluded from the replication. The last steps of the configuration process are: Subscribe the Publication, choosing if the replication is unidirectional or bidirectional, and Deploy the replication plan. After the Deploy, if the replication has started correctly, the line connecting the two c-tree servers turns green as shown in Figure 9, Replication Plan deployed, where the data on the FAIRCOMS server running on Windows are replicated on the FAIRCOMS server running on Ubuntu Linux.

Figure 9. Replication Plan deployed.



An overview of the current replication status and total amount of commits, records added, updated or deleted, can be activated from the Activity menu item of the pop-up menu of the deployed plan, as shown in Figure 10, *Replication Activity*.

Figure 10. Replication Activity.

Plan activity for plan : RepPlan1																							
Options																							
Time Frame:		Start Date: 12/16/2022		End Date: 12/16/2022		Start Time: 11:17 AM		End Time: 11:17 PM		Auto Display Refresh Rate: 0 sec.													
Details																							
Statistics Exceptions Function Timing																							
Snapshot Status : Running Stop																							
Identification		Replication				Thread Counts			Transaction Throughput														
Type	Time	Paused	Aborted	Failures	Latency	Analyze	Apply	Commits	Add	Update	Delete												
20	sourceToTarget 12/16/2022 12:15:20 PM	0	0	0	0	0	8	3	2	0	1	*											
21	sourceToTarget 12/16/2022 12:15:30 PM	0	0	0	0	1	8	3	2	0	1												
22	sourceToTarget 12/16/2022 12:15:40 PM	0	0	0	0	1	8	3	2	0	1												
23	sourceToTarget 12/16/2022 12:15:51 PM	0	0	0	0	1	8	3	2	0	1												
24	sourceToTarget 12/16/2022 12:16:01 PM	0	0	0	0	1	8	3	2	0	1												
25	sourceToTarget 12/16/2022 12:16:11 PM	0	0	0	0	1	8	3	2	0	1												
26	sourceToTarget 12/16/2022 12:16:21 PM	0	0	0	0	1	8	3	2	0	1												
27	sourceToTarget 12/16/2022 12:16:31 PM	0	0	0	0	1	8	3	2	0	1												
28	sourceToTarget 12/16/2022 12:16:41 PM	0	0	0	0	1	8	3	2	0	1												

## isCOBOL Compiler

The isCOBOL 2023 R1 compiler includes new compiler options to simplify the migration from other COBOL dialects and ESQL pre-compilers. It also supports a new syntax to declare a variable number of parameters in the program and class.

### Improved Compatibility with other COBOLs

In this release the ON statement has been implemented to allow execution of statements based on a counter that is implicitly defined for any ON statement. The counter is initialized to zero and is automatically increased by one every time the ON statement is executed. It is particularly useful to execute code depending on the number of CALLs executed on a program or the PERFORMs executed on a paragraph without declaring data items and manually manage the counter. For example, the following code snippet:

```
perform my-paragraph 10 times.  
...  
my-paragraph.  
  on 1           | "loop started first time"  
    initialize w-data.  
  ...  
  on 2 and every 2      | "execute at 'even' times"  
    move w-data to w-data-p  
  else  
    move w-data to w-data-e  
  .
```

executes the initialize statement only the first time the paragraph “my-paragraph” is executed and the move statement is done on different data items depending on the execution being odd or even.

MicroFocus COBOL and IBM COBOL allow specifying an occurs data item without the index after the data name, like the following code snippet:

```
77 w-flag pic 9 occurs 3.  
...  
  if w-flag = 1  
  ...
```

Previous isCOBOL compilers would have marked the statement as a Severe Error:

--S: #41 Subscript required: W-FLAG

To improve compatibility, when compiling using options -cm (for MicroFocus compatibility) or -cv (for IBM compatibility), the severe errors are now marked as managed errors, as shown below:

--E: #299 Subscript required, first occurrence assumed: W-FLAG

causing the program to be compiled successfully, and assuming that the data name W-FLAG is considered as W-FLAG(1).

New compiler options have been added in this release:

- **-csdb2** to activate the DB2 pre-compiler compatibility. With this option, all the behaviors activated by previous configuration `iscobol.compiler.esql.db2=true` are applied, for example the compiler generates specific code to return the result sets in the same format that would be produced when using the IBM DB2 preprocessor. It supports the SQLDA structure and the use of date, time and timestamp as function parameters, and it allows you to intercept the result of a function or a special register with a SET statement.
- **-csora** to activate the Oracle ProCobol compatibility. With this option, the Compiler enables the SQLADR, SQLNUL and SQLPRC functions used by SQLDA feature for the JDBC environment instead of the ProCobol native calls. An additional advantage of these new compiler options related to ESQL is that they are integrated and taken into consideration by the `iscobol.runtime.compile_flags.mandatory` or `iscobol.runtime.compile_flags.prohibited` configurations set when running.
- **-dcv** to use the VAX/COBOL numeric formats. These formats are identical to the IBM formats, except that unsigned COMP-3 fields place X"0C" in the sign position, instead of X"0F", useful when migrating data from VAX/COBOL or OpenVMS systems.

## Improved Compatibility with IBM DB2 Preprocessor

Enhancements have been made in this release to improve compatibility with the IBM DB2 Preprocessor.

The new supported syntax includes:

- CONNECT RESET statement to close the connection with the database. This statement can be used as synonym of DISCONNECT regardless of the database you're connected to.
- SQL TYPE in host variables declaration to map the host variable with a large object field on the database. The new syntax supported by the compiler in USAGE is:

```
01 Data-Item USAGE IS SQL TYPE IS { BLOB } [(Lob-Length)]  
{ CBLOB }  
{ DBCLOB }
```

The compiler internally transforms the host variables into a group data item where data and length are stored in two separate sub-items, for example the following host variables:

```
01 MY-BLOB USAGE IS SQL TYPE IS BLOB(2M).
```

is translated to:

```
01 MY-BLOB.  
  49 MY-BLOB-LENGTH PIC S9(9) COMP-5.  
  49 MY-BLOB-DATA   PIC X(2097152).
```

The following code snippet reads the content of a CLOB column and displays it:

```
01 MY-CLOB USAGE SQL TYPE IS CLOB(1M).  
  ...  
  EXEC SQL  
    SELECT TBL_CLOB INTO :MY-CLOB FROM TBL WHERE TBL_PK = 1  
  END-EXEC  
  DISPLAY MY-CLOB-DATA(1:MY-CLOB-LENGTH).
```

Although this syntax was implemented as compatibility with DB2, it can also work on other JDBC-compliant databases, and the compiler accepts the syntax without the need to set a specific option.

A new runtime configuration option has been added:

```
iscobol.esql.db2.row_data_as_bytes_threshold
```

The DB2 Preprocessor allows you to store any character in CHAR and VARCHAR fields, as you can do with a COBOL item with picture X. Some COBOL applications take advantage of this possibility to store the dump of COBOL structures with COMP fields into CHAR and VARCHAR fields. When moving to Java JDBC, CHAR and VARCHAR fields are managed as strings and a conversion error may occur if an unknown character is detected in the string. The database encoding affects the management of the strings.

To address this problem, the isCOBOL runtime allows you to manage CHAR and VARCHAR fields as byte arrays, allowing any character to be stored into such fields. To have all CHAR and VARCHAR fields managed as byte arrays, it's now possible to set:

```
iscobol.esql.db2.row_data_as_bytes_threshold=1
```

To have only specific CHAR and VARCHAR fields managed as byte arrays, set the property to a value greater than 1. For example, by setting:

```
iscobol.esql.db2.row_data_as_bytes_threshold=100
```

CHAR and VARCHAR fields whose size is not less than 100 bytes will be managed as byte arrays, while smaller CHAR and VARCHAR fields will be managed as strings.

### Variable number of parameters

With previous releases, when a program-id or entry needed to be called with a variable number of parameters, it was necessary to declare many linkage data items used in the "procedure division using" statement and call the C\$NARG library routine to find out the number of parameters passed from the caller program. In addition, in cases of method-id declared in a class-id, the variable number of parameters was not supported.

Starting from 2023 R1 the syntax "..." written at the end of the object reference class definition is supported to manage a variable number of arguments in all scenarios (program-id, entry, method-id). The syntax is the same as pure Java, where the "..." is used to create a method with a variable number of arguments (known also as a Varargs method). This is shown in the following code snippet:

```
public static void doDisplay (String... params){  
    System.out.println("Number of arguments received: " + params.length);  
    for(String s:params){  
        System.out.println(s);  
    }  
}
```

The same result can be achieved in the class-id Cobol source using the following syntax:

```
method-id. doDisplay as "doDisplay".
working-storage section.
77 npar    pic 9(3).
77 idx     pic 9(3).
linkage section.
77 params  object reference "java.lang.String...".
procedure division using params.
main.
  set npar to params:>length.
  display "Number of arguments received: " npar.
  perform varying idx from 0 by 1 until idx = npar
    display params(idx)
  end-perform
end method.
```

The calling programs that need to invoke this method can pass a variable number of parameters as shown in the following code snippet:

```
repository.
  class tclassid as "tclassid"
  .

  working-storage section.
 77 obj object reference tclassid.
procedure division.
  set obj to tclassid:>new().
  obj:>doDisplay(var1).
  obj:>doDisplay(var1, var2).
  obj:>doDisplay(var1, var2, "string").
```

The same applies for a program-id Cobol source. To specify a variable number of pic X data items instead of java.lang.String, declare the isCOBOL class type as shown in this code snippet:

```
program-id. tprogid.
working-storage section.
77 npar    pic 9(3).
77 idx     pic 9(3).
linkage section.
77 params  object reference "com.iscobol.types.PicX...".
procedure division using params.
main.
  set npar to params:>length.
  display "Number of arguments received: " npar.
  perform varying idx from 0 by 1 until idx = npar
    display params(idx)
  end-perform
  goback.
```

The caller program can pass a variable number of parameters in the CALL statement as shown in the following code snippet:

```
working-storage section.  
77 var1    pic x(10).  
77 var2    pic x any length.  
procedure division.  
    call "tprogid" using var1  
    call "tprogid" using var1, var2  
    call "tprogid" using var1, var2, "string"
```

## IsCOBOL Runtime

The isCOBOL runtime 2023 R1 includes a new file handler named VisionJ to access Vision files without the need for native code. It also supports new library routines and improved existing routines and configurations.

### New file handler VisionJ

A new value is supported for iscobel.file.index to fully access Vision file formats from versions 3 to 6, with the exception of encrypted files, without the need to create the Vision File Connector based on native code. The short alias name is VisionJ, while the full class name that needs to be used in CLASS clause of SELECT of the file definition is com.iscobol.io.DynamicVisionJ. This file handler manages locks in conjunction with ACUCOBOL-GT runtime, making it a good solution for mixed production sites where there are still ACUCOBOL-GT applications running. The approach is similar to the Vision File Connector but with higher performance since there is no need of pipe communication between processes.

As shown Figure 11, *Comparing VisionJ vs Vfc*, performance comparison was made using the IO-INDEXED program provided in the sample/io-performance folder to test 100,000 records managed by new the VisionJ file handle and the existing Vision file connector, known also as VFC. The test was run on Windows 11 Pro 64-bit with i7-8550U CPU @ 1.80GHz and 16 GB of RAM and on Linux Fedora 37 64-bit with i7-9750H CPU @ 2.60Ghz x 12 and 32 GB of RAM.

Figure 11. Comparing VisionJ vs Vfc.

Statement	VFC - Windows	VisionJ - Windows	VFC - Linux	VisionJ - Linux
WRITE:	12.66	8.75	10.72	7.37
READ:	6.65	3.17	1.69	0.74
REWRITE:	7.76	4.59	3.46	2.31
DELETE:	9.74	6.13	5.98	5.42
<b>Total Time:</b>	<b>36.81</b>	<b>22.64</b>	<b>21.85</b>	<b>15.84</b>

These tests show that the new VisionJ handler is faster in all operations and especially in the READ access, where the improvement is more than 50%. The new file handler is also integrated in all Veryant products, including the isCOBOL File Server and isCOBOL UDBC driver as well as isCOBOL utilities such as GIFE and ISMIGRATE. Specific configurations can be set with the prefix iscobel.visionj. For example, to create Vision files version four, set iscobel.visionj.v\_version=4.

## New and improved library routines and configurations

New library routines have been implemented to improve compatibility with the MicroFocus dialect, such as:

- CBL\_GET\_KBD\_STATUS to know if there are characters available from the keyboard
- CBL\_READ\_KBD\_CHAR to retrieve the character that was typed, in ASCII
- CBL\_THREAD\_SLEEP to sleep the given number of milliseconds

The following code snippet shows the syntax and how to use these new library routines:

```
77 key-status  pic x comp-x.
77 wcharacter  pic x.

...
perform until wcharacter = x"1B" | ESC
  call "CBL_GET_KBD_STATUS" using key-status
  if key-status = 1
    call "CBL_READ_KBD_CHAR" using wcharacter
    ...
    call "CBL_THREAD_SLEEP" using 5
  end-if
end-perform
```

To simplify the string conversion from one encoding to another a new routine named `C$STRING_CONVERT` is now available. This is used in multiple instances; for example to prepare data to be passed to a web service or to convert data received from a third-party application in the charset used by the isCOBOL application. The following code snippet shows how to convert the Euro sign from Windows Cp1252 charset where the character is stored by 1 byte x"80" into UTF-8 charset where the same character is stored in 3 bytes x"E282AC":

```
77 w-string      pic x(10).
77 w-string-utf8  pic x(10).

...
  initialize w-string-utf8
  move "€" to w-string
  call "C$STRING_CONVERT" using w-string
                w-string-utf8
                "Cp1252"
                "UTF-8"
```

Library routines that refer to file names in parameters now support the use of pic N instead of pic X to correctly manage the files whose name includes characters that are not supported by the current encoding. For example, the following code snippet allows you to correctly retrieve the file name with Chinese characters even if the application is running on a Windows Latin-1 platform where the default encoding is CP1252:

```

copy "isopensave.def" replacing ==pic x== by ==pic n==.
77  file-name          pic n(256) .
01  file-info.
    03  file-size      pic x(8) comp-x.
    03  file-date       pic 9(8) comp-x.
    03  file-time       pic 9(8) comp-x.
77  ret-code          pic s9.

...
    initialize opensave-data
    call "C$OPENSAVEBOX" using opensave-open-box
                           opensave-data
                           giving ret-code
if ret-code = 1
    move opnsav-filename to file-name
    call "C$FILEINFO" using file-name
                           file-info
                           giving ret-code
...
end-if

```

In general, these library routines allow you to pass both pic X or pic N fields when the parameter is elementary, like the first parameter of C\$FILEINFO. In cases where a parameter is a structure 01 level included in a .def copy file, like the opensave-data for C\$OPENSAVEBOX, it's necessary to use the copy replacing syntax to ensure that the child levels in the structure defined in the copy file will also use a pic N syntax.

The W\$KEYBUF routine has been enhanced to support the Alt+letter/num combination key using {@?} syntax to pass a value from @A to @Z for letters or from @0 to @9 for numbers. For example, the following code snippet sends Alt+5 key to the next accept:

```

copy "iscobol.def".
77  w-comb-key  pic x(4) .
...
move "{@5}" to w-comb-key
call "W$KEYBUF" using wdbuf-add-to-end, w-comb-key

```

These are the new supported configurations options:

- *iscobol.ctree.bound\_library* to set the library name to be loaded in bound\_server mode, when *iscobol.ctree.bound\_server=true* is set. It has a default value of “ctdbsapp”, the correct value for c-treeRTG version 3.0.2.668 released with 2023R1. In previous versions of ctree the library was named “ctreedbs”. Set this configuration variable to “ctreedbs” if you wish to start an older c-tree version in bound mode.
- *iscobol.guiIMPLIED\_decimal* to have the implied\_decimal feature on both character accept and graphical entry-fields. When set to true, if no decimal separator is specified by the user, a decimal separator is automatically applied by the Runtime according to the picture of the data-item bound to the input field.

# GUI enhancements

Improvements to GUI controls have been implemented in this release, especially in the tree-view control and window container to optimize screen refresh.

## Enhancements on tree-view

The tree-view control now supports the VPADDING property to affect the height of the items, indicating extra vertical space to be applied to each item. The value of this property is expressed as a percentage of the control's font, like in the grid control. This property is supported by both the standard tree-view without columns and the Table-View with columns.

In the tree-view with Table-View style, the ability to display icons in the different columns through the BITMAP-NUMBER and BITMAP-TRAILING properties in the column identified by the X property is now supported.

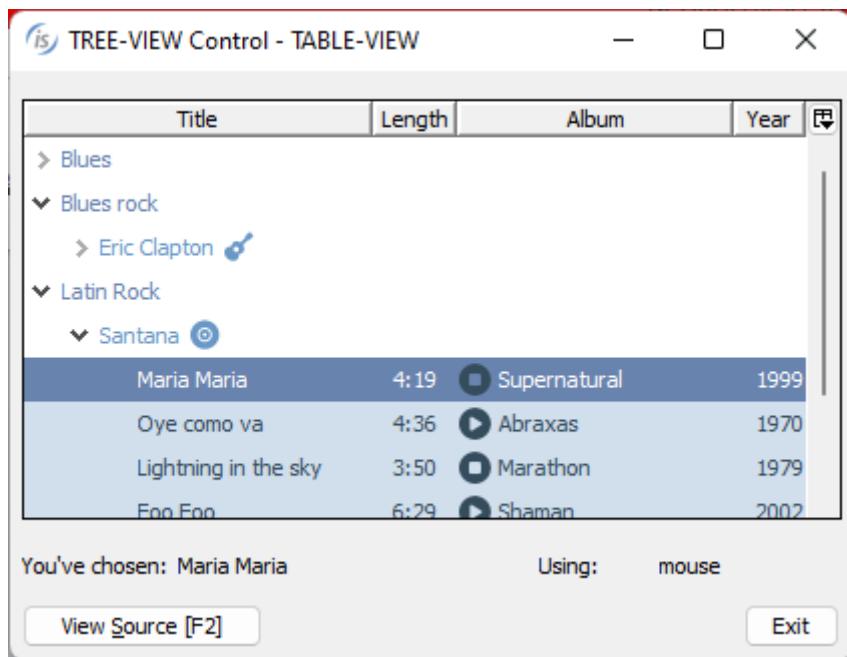
The following code snippet:

```
03 tree-table tree-view table-view
    bitmap-handle hBmpAC
    bitmap-width 16
    ...
    vpadding 50
    ...
    modify tree-table parent tv-item-parent item-to-add rec-grid
        giving tv-item-child has-children 1
        x 1 bitmap-number wrk-num-bmp bitmap-trailing 1
    modify tree-table parent tv-item-child item-to-add rec-grid
        giving tv-item has-children 0
        x 1 bitmap-number 0
        x 3 bitmap-number wrk-num-bmp-2
```

declares a tree-view control with table-view style and a bitmap strip during the loading.

Different icons are set on different items. Icons are set on the first column after the text and on the third column before the text since bitmap-trailing is not set, resulting in the default value 0 to be used. The result of the code is shown in Figure 12, *Table-View with icons*.

Figure 12. Table-View with icons.



## Window optimization

Typical graphical applications display complex forms, potentially containing hundreds of controls. Such applications can suffer for sub-optimal screen refresh performance, depending of the performed operation or implementation details.

Potential problematic areas include, for example, windows composed of multiple screens, windows where individual controls are created dynamically using DISPLAY statements, and windows that are destroyed and then re-created, causing content such as menu bar, tool-bar and ribbon controls to be recreated.

In these situations, depending on the device and monitor refresh performance, or network speed when running in ThinClient or WebClient, a screen refresh could be aesthetically unappealing and perform poorly.

To avoid this, the MASS-UPDATE property, already supported on controls such as grid, tree-view, list-box and combo-box, can be applied to a window to group multiple user interface updates in a single refresh. Just set MASS-UPDATE to 1 before a large UI refresh and set again MASS-UPDATE to 0 when the all updates are completed. All the changes are applied instantly, resulting in higher performance and a smoother and more pleasing user experience.

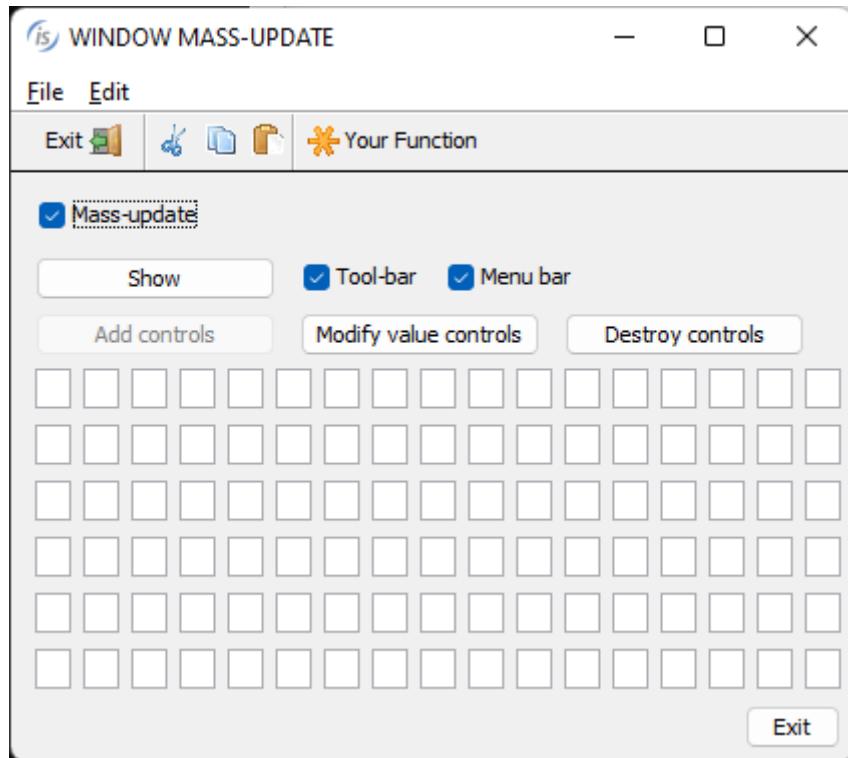
MASS-UPDATE on controls should still be used to ensure best performance.

The following code snippet take advantage of the new implementation:

```
modify hWin mass-update 1
destroy Mask
perform DESTROY-MENU
perform DESTROY-TOOLBAR
perform SHOW-MENU
perform DISPLAY-TOOLBAR
display Mask
perform until ...
...
    display entry-field handle ef-handle(idx)
        line w-d-line col w-d-col size 3 cells
end-perform
modify pb-add-control      enabled e-pb-add-control
modify pb-modify-control  enabled e-pb-modify-control
modify pb-destroy-control enabled e-pb-destroy-control
modify hWin mass-update 0
```

The result of the code is shown in Figure 13, *Mass-update on window*. When debugging the program, it's possible to note that after stepping on the "modify hWin mass-update 1", all the statements that affect controls do not refresh the screen, and only after executing "modify hWin mass-update 0" the screen is fully refreshed.

Figure 13. Mass-update on window.



## isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. As of version isCOBOL 2023 R1, the `HTTPClient` class can set a Proxy server to be used when performing network activity, and a new specific log file has been implemented to log the activity.

### HTTPClient class

`HTTPClient` is a class that enables COBOL programs to interact with Web Services and HTTP Servers. This class has been updated with this new method signature:

```
public void setProxy (String Ip, Int Port)
```

After invoking this method, the specified proxy server is used for all the following requests performed by the `HTTPClient` instance.

New configurations have been implemented to trace the `HTTPClient` activity:

- `iscobol.httpclient.logging=true` to enable logging
- `iscobol.httpclient.logfile=/path/to/logfile` to specify the log pathname

The log includes the request date and time, the request content, the response code, the response headers and the response content. If the file already exists, the new content is appended to it.

This is an example of the log content running the installed sample IP2GEO with the two new configuration settings:

```
=====
Connection requested at 2022-12-28 - 17:28:07.287
Connecting to: http://ws.cdyne.com/ip2geo/ip2geo.asmx
Request content:
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope xmlns:soapenv="http://
www.w3.org/2003/05/soap-envelope" xmlns:tns="http://ws.cdyne.com/
"><soapenv:Body><tns:ResolveIP><tns:ipAddress>209.235.175.10</
tns:ipAddress><tns:licenseKey/></tns:ResolveIP></soapenv:Body></soapenv:Envelope>

Response code: 200
Response Headers:
HTTP/1.1 200 OK
Cache-Control:no-cache
Pragma:no-cache
Content-Type:application/soap+xml; charset=utf-8
Expires:-1
Server:Microsoft-IIS/10.0
X-AspNet-Version:4.0.30319
X-Powered-By:ASP.NET
Date:Wed, 28 Dec 2022 16:28:06 GMT
Content-Length:658
Response content:
<?xml version="1.0" encoding="utf-8"?><soap:Envelope xmlns:soap="http://www.w3.org/
2003/05/soap-envelope" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"><soap:Body><ResolveIPResponse
xmlns="http://ws.cdyne.com/"><ResolveIPResult><City>Nashville</
City><StateProvince>TN</StateProvince><Country>United States</Country><Organization />
<Latitude>36.1114</Latitude><Longitude>-86.869</Longitude><AreaCode>615</
AreaCode><TimeZone /><HasDaylightSavings>false</HasDaylightSavings><Certainty>90</
Certainty><RegionName /><CountryCode>US</CountryCode></ResolveIPResult></
ResolveIPResponse></soap:Body></soap:Envelope>
=====
Connection requested at 2022-12-28 - 17:39:21.3921
Connecting to: http://ws.cdyne.com/ip2geo/ip2geo.asmx
Request content:
...
```

## Additional improvements

isCOBOL 2023 R1 release improves the JUTIL and Stream2Wrk utilities.

### JUTIL

JUTIL is the ISAM utility for indexed files in JISAM format. In this release, two existing options have been enhanced providing additional parameters.

The -convert option supports two additional optional parameters:

- *-d* to delete the intermediate files after converting
- *-s* to strip the extension from the output file name

These options are useful to avoid double extensions in the migrated JISAM file, for example when running the following command:

```
jutil -convert products.ext out-jisam -d -s
```

the MicroFocus indexed file named products.ext is converted to JISAM format. The JISAM file is created in the out-jisam folder, the physical file name is products.dat and products.idx, since the -s parameter has been passed. Without passing the -s parameter, the physical file name is products.ext.dat and products.ext.idx. The extension .dat and .idx are the defaults for JISAM, but they can be customized by configuring iscobel.file.index.data\_suffix and iscobel.file.index.index\_suffix. Temporary files are created in the user's temporary directory unless the TMPDIR environment variable is set, and are deleted after the conversion as a result of specifying the -d parameter.

The –rebuild option supports an additional optional parameter:

- *-efd=efdfile.xml* to create a new idx file based on the xml information.

This is useful when the index file is corrupted in the header information or completely missing, and to correctly rebuild the JISAM file it's important to create first the appropriate empty .idx file. This process is now transparent when passing the .xml file created by -efd compiler option. For example, the following command:

```
jutil -rebuild products -efd=xml
```

will rebuild the JISAM file named products when only the .dat file is present, and the .idx is missing. The .idx file will be created using the file info from products.xml file stored in the xml folder.

## Stream2Wrk

Stream2Wrk is a utility that developers can use to generate copy files containing the data structures to be used in working-storage section when parsing a stream. In addition to the supported wsdl, xml and json format, now it's possible to parse a .xsd file directly, without the need for an .xml file that implements the xsd structure. For example, the following command:

```
stream2wrk xsd order.xsd
```

parses the order.xsd file and creates an order.wrk copybook.

The xsd option supports the same parameters as xml parsing:

- *[-o outputfile]* to set a different output file
- *[-p prefix]* to set a prefix to be used in the generated data-names
- *[-d]* to activate the disambiguate rule in order to avoid ambiguous identifiers

In addition, new optional parameters have been implemented in both xml and xsd parsing:

- *[-c]* to generate 'count' data-items
- *[-e]* to generate 88 level representing 'enumeration' tags
- *[-iu]* to ignore unbounded, avoiding the generation of "occurs" when there is 'maxOccurs=unbounded'
- *[-l[=len]]* to generate data-items with fixed size instead of 'pic x any length'
- *[-nc]* to avoid the generation of commented lines
- *[-sa attribute-suffix]* to specify the suffix for 'attribute' data-items
- *[-sc count-suffix]* to specify the suffix for 'count' data-items

- *[-scp capacity-suffix]* to specify the suffix for 'capacity' data-items when an "occurs dynamic" is declared
- *[-sd data-suffix]* to specify the suffix for 'data' data-items
- *[-se enumeration-suffix]* to specify the suffix for enumeration data-items

These new options help COBOL developers to better customize the generated output, and some of them are necessary if the generated copybook needs to be used in an XD file definition, where pic x any length and occurs dynamic are not supported. Occurs dynamic and pic x any length are still preferable when using the XMLStream class.

For example, parsing the following xsd file:

```

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:tns="http://tempuri.org/PurchaseOrderSchema.xsd"
              targetNamespace="http://tempuri.org/PurchaseOrderSchema.xsd"
              elementFormDefault="qualified">
  <xsd:element name="PurchaseOrder" type="tns:PurchaseOrderType"/>
  <xsd:complexType name="PurchaseOrderType">
    <xsd:sequence>
      <xsd:element name="ShipTo" type="tns:USAAddress" maxOccurs="2"/>
    </xsd:sequence>
    <xsd:attribute name="OrderDate" type="xsd:date"/>
  </xsd:complexType>
  <xsd:complexType name="USAAddress">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="state" type="xsd:string"/>
      <xsd:element name="zip" type="xsd:integer"/>
      <xsd:element name="priority">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
            <xsd:enumeration value="high"/>
            <xsd:enumeration value="normal"/>
            <xsd:enumeration value="low"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

with the following command:

```
stream2wrk xsd order.xsd -o order1.wrk -p order- -d
```

the generated order1.wrk file will contain the following structure:

```
01 order-PurchaseOrder identified by 'PurchaseOrder'
    namespace 'http://tempuri.org/PurchaseOrderSchema.xsd'.
03 order-OrderDate-attr identified by 'OrderDate'
    is attribute pic x any length.
03 order-ShipTo identified by 'ShipTo' occurs 2.
05 order-name identified by 'name'.
07 order-name-data pic x any length.
05 order-state identified by 'state'.
07 order-state-data pic x any length.
05 order-zip identified by 'zip'.
07 order-zip-data pic s9(18).
05 order-priority identified by 'priority'.
07 order-priority-data pic x any length.
```

while using the additional parameters with the following command:

```
stream2wrk xsd order.xsd -o order2.wrk -p order- -d -e -c -iu -l=80 -sd      -var -se -88
```

the generated order2.wrk file contains this structure:

```
01 order-PurchaseOrder identified by 'PurchaseOrder'
    namespace 'http://tempuri.org/PurchaseOrderSchema.xsd'
    count in order-PurchaseOrder-count.
03 order-OrderDate-attr identified by 'OrderDate'
    is attribute pic x(80) count in order-OrderDate-attr-
count.
03 order-ShipTo identified by 'ShipTo' occurs 2 count in order-ShipTo-
count.
05 order-name identified by 'name' count in order-name-count.
07 order-name-var count in order-name-var-count pic x(80).
05 order-state identified by 'state' count in order-state-count.
07 order-state-var count in order-state-var-count pic x(80).
05 order-zip identified by 'zip' count in order-zip-count.
07 order-zip-var count in order-zip-var-count pic s9(18).
05 order-priority identified by 'priority' count in order-priority-
count.
88 order-priority-88-0 value 'high'.
88 order-priority-88-1 value 'normal'.
88 order-priority-88-2 value 'low'.
07 order-priority-var count in order-priority-var-count pic x(80).
```

# isCOBOL Evolve

---

## isCOBOL 2022 Release 2 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2022 R2.

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

The new compiler supports an integrated PreProcessing feature and increased compatibility and performances with other COBOL dialects.

The new WebClient release now supports Java 17 and offers several improvements and higher performance on slow networks.

isCOBOL Evolve 2022 R2 GUIs are now Hi-DPI aware and include improvements, refinements, and new compatibility features to simplify the migration of existing COBOL programs to the isCOBOL Evolve suite.

Details on these enhancements and updates are included below.

### isCOBOL Compiler

The isCOBOL 2022 R2 compiler includes a new feature that allows developers to write a PreProcessor in COBOL or in Java to integrate custom logic to be applied to source code.

Compatibility with other COBOL dialects and performances has been improved to simplify migrations, and isCOBOL ESQL has been improved to ease migration from IBM DB2 PRE and Oracle Pro\*COBOL (ESQL precompilers).

#### PreProcessor integration

A new compiler configuration, "iscobol.compiler.custompreproc", has been added to enable Preprocessor parsing. Set this configuration to one or more classes that implement the isCOBOL compiler interface "com.iscobol.compiler.custpreproc.LinePreProcessor". For example, when compiling a program with this set in the configuration:

```
iscobol.compiler.custompreproc=MyPreProc SecondPreProc
```

the compiler will pass every line of the source code being compiled to the MyPreProc class, which may perform source code changes. The resulting line is then passed to the SecondPreProc class, which can perform additional changes, and the resulting source code line is finally passed to the compiler for compilation.

isCOBOL already supports the use of the compiler option "iscobol.compiler.regexp" that can be used to replace text within the source code, but the new PreProcessor feature is more flexible and allows the parsing of multiple lines of code for analysis.

Both configuration options can be used simultaneously, letting you make complex code substitutions. The regexp will be executed first, then the custom preprocessor.

To implement the com.iscobol.compiler.custpreproc.LinePreProcessor interface, the class needs to include a "process" method with the signature:

```
public void process(java.lang.String originalLine,
                    int sourceFormat,
                    java.lang.String fileName,
                    int lineNumber,
                    java.lang.String[] compilerOptions,
                    com.iscobol.compiler.custpreproc.ProcessResult result)
    throws com.iscobol.compiler.custpreproc.ProcessException
```

So, the source can be written in COBOL declaring a CLASS-ID that has a METHOD-ID receiving these parameters:

```
class-id. MyPreProc as "MyPreProc" implements LinePreProcessor.
repository.
class LinePreProcessor as "com.iscobol.compiler.custpreproc.LinePreProcessor"
class ProcessResult as "com.iscobol.compiler.custpreproc.ProcessResult"
class ProcessException as "com.iscobol.compiler.custpreproc.ProcessException"
...
method-id. process as "process".
linkage section.
77 original-line object reference "java.lang.String".
77 source-format object reference "int".
77 source-file-name object reference "java.lang.String".
77 line-number object reference "int".
77 compiler-options object reference "java.lang.String[]".
77 result object reference ProcessResult.
procedure division using original-line,
                     source-format,
                     source-file-name,
                     line-number,
                     compiler-options,
                     result
                     raising ProcessException.
```

For example, this method can be used to implement custom logic to write comments for a block of code, or to replace some code with another. You can also return Warnings or Errors exactly like the standard compiler produces. Another example use of the PreProcessor is to enforce or suggest code changes to comply with a desired standard set of rules.

When compiling sources with the -If option, the resulting .list file will contain the output after the processing executed by the preprocessor classes is finished.

When debugging a COBOL program compiled using the preprocessor, the source shown will be the original source, as written by the developer, but the instructions executed will be the result of the preprocessor output.

To better follow the logic of the class-id MyPreProc, you can debug the COBOL source with the Remote Debugger feature. To do that, first compile with this variable set in the configuration:

```
iscobol.compiler.rundebug=2
```

and start the Remote Debugger with the command:

```
iscrun -d -r
```

This will start the Remote Debugger on the default localhost using the default port number 9999 where the compilation process is waiting for the debugger execution. If the compilation is executed on a different server, you can pass the hostname of that server when launching the Remote Debugger.

For additional details on this feature, refer to the updated 2022 R2 documentation and new samples installed in the sample folder “compiler-pre-process”.

### Option to generate optimized class

isCOBOL 2022 R2 provides a new compiler option named “-zmf” that internally activates a new compiler engine designed to maximize runtime performance and minimize resource usage. The new compiler engine has been developed for maximum compatibility with Enterprise IBM Cobol and Micro Focus COBOL. This early version can be used for batch COBOL programs without user interface. There are some syntax limitations as well in this first version.

The new compiler engine uses top-down (recursive descent) parsers as opposed to bottom-up parsers generated by a YACC-like tool. Based on the most popular parser generator, JavaCC, the new compiler engine runs in three steps:

1. COBOL syntax validation
2. Walk to collect data and to generate functions
3. Emitters, LAMBDA based to emit Java classes

As shown Figure 1, *NCR CSAMPLE Benchmark*, performance comparison was made using NCR CSAMPLE benchmark test between isCOBOL 2022 R2 with “-zmf” and other COBOL compilers from the market. The test was run on Windows 11 Pro 64-bit with i7-8550U CPU @ 1.80GHz and 16 GB of RAM. The test shows that isCOBOL’s new compiler engine beat our competitors by running 13% - 80% faster and executing at least 22% more instructions per second.

Figure 1. NCR CSAMPLE Benchmark.

Statement	isCOBOL-zmf	COBOLA	COBOL B	COBOL C	COBOLD
MOVE	0.875	1.191	0.729	1.179	0.967
ADD	0.078	1.166	0.788	1.755	0.351
CALL	0.040	0.370	0.170	2.560	0.030
COMPUTE	0.003	0.035	0.005	0.020	0.023
DIVIDE	0.244	0.561	0.514	0.227	0.028
GO TO	0.021	0.038	0.016	0.073	0.020
IF	0.049	0.115	0.069	0.201	0.100
INSPECT	0.070	0.930	0.620	0.230	0.090
MULTIPLY	0.064	0.934	0.913	0.317	0.264
PERFORM	0.021	0.566	0.126	0.743	0.045
SEARCH	0.020	0.480	0.400	0.120	0.080
SET INDEX	0.003	0.082	0.065	0.066	0.033
STRING	0.022	0.244	0.229	0.258	0.030
SUBTRACT	0.003	0.020	0.003	0.040	0.007
UNSTRING	0.097	0.598	0.654	0.479	0.142
<b>Total</b>	<b>1.610</b>	<b>7.330</b>	<b>5.301</b>	<b>8.268</b>	<b>2.210</b>
<b>KIPS (Kilo Instructions Per Second)</b>	<b>148,810</b>	<b>55,160</b>	<b>117,316</b>	<b>32,904</b>	<b>111,086</b>

## Improved Compatibility with other COBOLs

Enhancements have been implemented in this release to improve compatibility with MicroFocus COBOL, such as:

- support for THREAD-LOCAL-STORAGE SECTION and IS THREAD-LOCAL clauses.

The THREAD-LOCAL-STORAGE Section describes data which is unique to each thread, useful for resolving contention problems related to concurrent access to the same dataitem. If a data-item is declared inside this new SECTION, every thread has its own copy of the variable instead of sharing the same with other threads.

The IS THREAD-LOCAL clause can be used in the WORKING-STORAGE SECTION or on the FD level to declare that a data-item or File Record needs to be treated with the thread logic described above, making every thread access its separate variables or record definitions.

Here is a code snippet that shows how to declare variables in the THREAD-LOCALSTORAGE SECTION and variables and FD with the THREAD-LOCAL clause:

```

file section.
fd file-prod is thread-local.
01 prod-rec.
  03 prod-key pic 9(4).
  03 prod-desc pic x(90).
working-storage section.
77 thr1 handle of thread.
77 thr2 handle of thread.
77 counter pic 9(4) value 0 is thread-local.
thread-local-storage section.
01 w-rec.
  03 w-key pic 9(4).
  03 w-desc pic x(90).

```

- new compiler option -smfu to manage the Underscore in sources with MF compatibility

With isCOBOL, underscore and hyphen are treated as the same character, but in the MicroFocus compiler the underscore and hyphen are 2 different characters. When a COBOL program contains the same variable or paragraph name, with the only difference being an underscore and hyphen, the isCOBOL compiler will throw a Severe "Ambiguous identifier" and "Duplicate procedure name" error message when compiling.

With the new compiler option –smfu, isCOBOL treats underscore and hyphen as 2 different characters, allowing sources as described to be compiled without errors.

For example, the following code can be compiled correctly:

```
working-storage section.  
77 my-var pic x.  
77 my_var pic x.  
...  
procedure division.  
...  
    move "a" to my-var.  
    move "b" to my_var.  
    perform SHOW-MY-VAR.  
    perform SHOW-MY_VAR.  
SHOW-MY-VAR.  
    display "my-var=" my-var.  
SHOW-MY_VAR.  
    display "my_var=" my_var.
```

- the semicolon is now accepted by the compiler as universal path separator

This simplifies sharing file paths between operating systems. Windows uses the ";" as a path separator, while MacOS and Linux use ":". The isCOBOL compiler will translate the universal path separator, ";", to the OS-specific character.

For example, the following command used to compile on Windows:

```
iscc -sp=copylib;other-copy/cpy program-name
```

can now be used without changes when compiling in Linux or MacOS.

This change also benefits the isCOBOL IDE when sharing the same Workspace to compile programs using different operating systems such as Windows, Linux and MacOS.

Enhancements have been made in this release to improve compatibility with IBM COBOL, such as:

- support for JSON PARSE and JSON GENERATE statements.

The existing XML PARSE and XML GENERATE statements gave you the ability to manage XML streams, and starting from this release, the JSON PARSE and JSON GENERATE statements are supported to manage JSON streams.

isCOBOL already supports the “com.iscobol.rts.JSONStream” class to manage JSON streams, but now direct migration from IBM COBOL sources that contain these statements is supported by the compiler. Here’s a code snippet of the instructions:

```
working-storage section.  
...  
01 Grp.  
  05 Acc-No pic AA9999.  
  05 More.  
    10 code pic S99V9 occurs 2.  
  05 cr-date pic 99/99/9999.  
01 json-stream pic x(80).  
01 i binary pic 99.  
procedure division.  
...  
  initialize grp  
  move "AB1234" to acc-no  
  move 1.2 to code(1)  
  move -3 to code(2)  
  move "01/07/2022" to cr-date  
  JSON GENERATE json-stream from grp  
    count i  
    name of code is 'Value'  
    suppress cr-date  
    on exception  
      display "Error on JSON generate"  
    not on exception  
      display "Successful JSON generate"  
  end-json.  
  
  initialize grp  
  JSON PARSE json-stream into grp  
    with detail  
    name of code is 'Value'  
  end-json.
```

The generated JSON will be:

```
{ "Grp" : { "Acc-No" : "AB1234", "More" : { "Value" : [1.2, -3.0] } } }
```

When the COBOL program receives a JSON stream, the JSON PARSE statement will parse the stream and set the values in the corresponding WORKING STORAGE items.

### Improved Compatibility with other ESQL Precompilers

Enhancements have been made in this release to improve compatibility with RDBMS PreCompilers such as IBM DB2 PRE and Oracle Pro\*COBOL.

- Support for Common Table Expressions (CTE)

A Common Table Expression (CTE) is the result set of a query which exists temporarily and for use only within the context of a larger query. The result of a CTE is not stored and exists only for the duration of the query. CTEs enable users to easily write and maintain complex queries with increased readability and simplification.

This reduction in complexity is achieved by deconstructing ordinarily complex queries into simple blocks to be used and reused as necessary in rewriting the query. CTEs initiate with the WITH keyword. Here's a code snippet as an example:

```
EXEC SQL
DECLARE CUR CURSOR FOR
WITH Sales_CTE AS
( SELECT SalesPersonID, SalesOrderID, YEAR(OrderDate) AS SalesYear
  INTO :wrk-person-id, :wrk-order-id, :wrk-year
    FROM Sales.SalesOrderHeader
   WHERE SalesPersonID IS NOT NULL )
SELECT SalesPersonID, COUNT(SalesOrderID) AS TotalSales, SalesYear
  FROM Sales_CTE
 GROUP BY SalesYear, SalesPersonID
 ORDER BY SalesPersonID, SalesYear
END-EXEC.
```

Note that some databases might not support this syntax, so ensure it's supported by your RDBMS before using. The larger RDBMS like IBM DB2, Oracle, Postgres, and Microsoft SQL Server support it.

- Ability to specify stored procedure and function names through host variables

Previously, the ESQL CALL statement required the procedure or function name to be a constant string. Now in 2022 R2 the name can be specified using host variables. For example:

```
MOVE "myproc" TO WRK-PROC-NAME.
EXEC SQL
  CALL :wrk-proc-name (:wrk-param-1 IN, :wrk-param-2 IN)
                INTO :wrk-result
END-EXEC.
```

This feature lets you write more dynamic code.

- Support for Oracle Pro\*COBOL SQLDA structure

The compatibility with ESQL preprocessors has been increased in this version by supporting the Oracle Pro\*COBOL format of the SQLDA structure. An SQLDA (SQL Descriptor Area) is a set of group items that store all the information the database needs about select-list items or place-holders for bind variables, except their values.

Previously, SQLDA was supported only in compatibility with the IBM DB2 preprocessor.

From the new isCOBOL 2022 R2 release, SQLDA is also supported with Oracle Pro\*COBOL.

The SQLDA structure from Oracle Pro\*COBOL is defined as follows:

```
01 SELDESC.  
02 SQLDNUM PIC S9(9) COMP VALUE 100.  
02 SQLDFND PIC S9(9) COMP.  
02 SELDVAR OCCURS 100 TIMES.  
03 SELDV PIC S9(9) COMP.  
03 SELDFMT PIC S9(9) COMP.  
03 SELDVLN PIC S9(9) COMP.  
03 SELDFMTL PIC S9(4) COMP.  
03 SELDVTYP PIC S9(4) COMP.  
03 SELDI PIC S9(9) COMP.  
03 SELDH-VNAME PIC S9(9) COMP.  
03 SELDH-MAX-VNAMEL PIC S9(4) COMP.  
03 SELDH-CUR-VNAMEL PIC S9(4) COMP.  
03 SELDI-VNAME PIC S9(9) COMP.  
03 SELDI-MAX-VNAMEL PIC S9(4) COMP.  
03 SELDI-CUR-VNAMEL PIC S9(4) COMP.  
03 SELDFCLP PIC S9(9) COMP.  
03 SELDFCRCP PIC S9(9) COMP.  
01 XSELDI.  
03 SEL-DI OCCURS 100 TIMES PIC S9(4) COMP.  
01 XSELDIVNAME.  
03 SEL-DI-VNAME OCCURS 100 TIMES PIC X(80).  
01 XSELDV.  
03 SEL-DV OCCURS 100 TIMES PIC X(80).  
01 XSELDHVNAME.  
03 SEL-DH-VNAME OCCURS 100 TIMES PIC X(80).  
01 BNDDSC.  
02 SQLDNUM PIC S9(9) COMP VALUE 100.  
02 SQLDFND PIC S9(9) COMP.  
02 BNDDVAR OCCURS 100 TIMES.  
03 BNDDV PIC S9(9) COMP.  
03 BNDDFMT PIC S9(9) COMP.  
03 BNDDVLN PIC S9(9) COMP.  
03 BNDDFMTL PIC S9(4) COMP.  
03 BNDDVTYP PIC S9(4) COMP.  
03 BNDDI PIC S9(9) COMP.  
03 BNDDH-VNAME PIC S9(9) COMP.  
03 BNDDH-MAX-VNAMEL PIC S9(4) COMP.  
03 BNDDH-CUR-VNAMEL PIC S9(4) COMP.  
03 BNDDI-VNAME PIC S9(9) COMP.  
03 BNDDI-MAX-VNAMEL PIC S9(4) COMP.  
03 BNDDI-CUR-VNAMEL PIC S9(4) COMP.  
03 BNDDFCLP PIC S9(9) COMP.  
03 BNDDFCRCP PIC S9(9) COMP.  
01 XBNDDI.  
03 BND-DI OCCURS 100 TIMES PIC S9(4) COMP.  
01 XBNDDIVNAME.  
03 BND-DI-VNAME OCCURS 100 TIMES PIC X(80).  
01 XBNDDV.  
03 BND-DV OCCURS 100 TIMES PIC X(80).  
01 XBNDDHVNAME.  
03 BND-DH-VNAME OCCURS 100 TIMES PIC X(80).
```

These data items are used in DESCRIBE, OPEN and FETCH ESQL statements, for example:

```
EXEC SQL DESCRIBE BIND VARIABLES FOR Stmt1 INTO BNDDSC END-EXEC.  
EXEC SQL DESCRIBE SELECT LIST FOR Stmt1 INTO SELDSC END-EXEC.  
EXEC SQL OPEN Curi USING DESCRIPTOR BNDDSC END-EXEC.  
EXEC SQL FETCH Curi USING DESCRIPTOR SELDSC END-EXEC.
```

## GUI enhancements

Many improvements to GUI controls have been implemented in this release. The isCOBOL framework is now fully dpi-aware. New properties and events are supported on several controls to enhance and improve GUI applications.

### Fully dpi-aware

High DPI screens are common, and DPI scaling is essential to ensure the correct scaling of graphical screens across devices.

High DPI screens have higher pixel density compared to regular screens, and not scaling correctly could result in screen fonts being too small to be readable, or just not looking good. Applications are defined as DPI-aware if they can scale fonts and images to maintain the correct aspect-ratio and quality across devices.

When running an application in Windows, for example, if the application is not dpi-aware (known also as dpi-unaware), the operating system automatically manages the dpi scaling, but causes quality loss as a result of the upscaling. Starting from isCOBOL 2022 R2, the framework is dpi-aware and can scale to maintain a good quality output.

For example, running the Samples menu program on a 144 DPI display (a Windows system with 150% character size scaling) with isCOBOL 2022 R1 or previous versions, the screen looked like Figure 2, *Samples menu program not dpi-aware*.

Figure 2. Samples menu program not dpi-aware

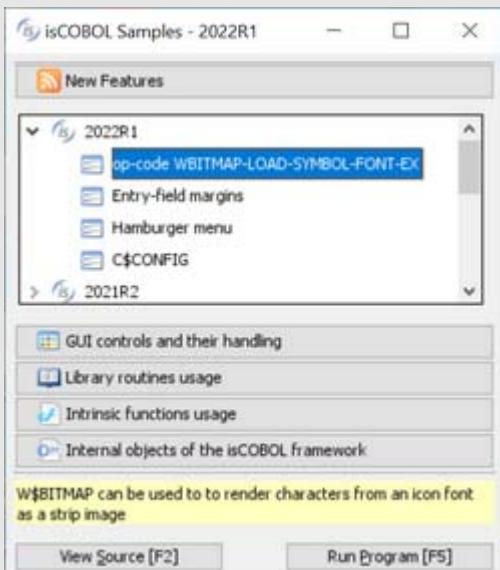
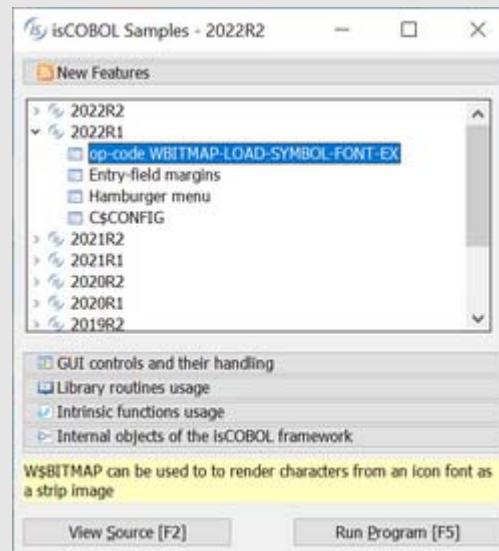


Figure 3. Samples menu program dpi-aware



Running the same sample program on the same 168 DPI display with isCOBOL 2022 R2, the screen looks like Figure 3. *Samples menu program dpi-aware*.

Previous isCOBOL versions used a virtualized dpi-aware approach, increasing the pixel representation and losing quality, while the new isCOBOL version uses a real dpi-aware scaling, increasing the font sizes to keep a high-quality look. The same applies to images and icons.

If the previous virtualized dpi-aware approach is needed, it can be restored on Windows systems as follows: locate the isrun.exe (or the executable file used to start the program), right click on it and choose the "Properties" menu item, navigate to the "Compatibility" tab, click on "Change high DPI settings", check the "Override high DPI scaling behavior" and choose "System" for "Scaling performed by:".

## New controls features

Several graphical controls that have been improved in the latest release:

- Check-Box and Radio-Button controls are now affected by a new configuration option named *iscobol.gui.icons\_scaling*. When it is set to true, resizing by using the LMZOOM layout-manager automatically scales the icons of the screen's Check-Box and Radio-Button controls.
- Check-Box, Radio-Button and Push-Button controls now support the BITMAP-SCALE property, previously supported only on bitmap controls. When using bitmaps on the check-box, radio-button, and push-button controls with the new BITMAP-SCALE property set to 1, the bitmap is resized along with the window when using the LMZOOM layout manager. This property affects all the bitmaps that are defined in Bitmap-Default, Bitmap-Disabled, Bitmap-Disabled-Selected, Bitmap-Number, BitmapPressed, Bitmap-Rollover, Bitmap-Rollover-Selected and Bitmap-Selected for controls that have only a bitmap, or both bitmap and text with the position defined in the TitlePosition property. A code snippet of controls with the new property is shown below.

```
03 check-box ...
  bitmap bitmap-handle bmp-handle bitmap-width 19
  bitmap-number 1 bitmap-rollover 2
  bitmap-scale 1.
03 push-button ...
  bitmap bitmap-handle bmp-handle bitmap-width 19
  bitmap-number 3 bitmap-rollover 4
  bitmap-scale 1.
03 radio-button ...
  bitmap bitmap-handle bmp-handle bitmap-width 19
  bitmap-number 5 bitmap-rollover 6
  bitmap-scale 1.
```

- Check-Box supports two new properties named CHECK-ON-VALUE and CHECK-OFFVALUE allowing you to specify alphanumeric values associated with checked and unchecked states. This simplifies the management of check-boxes that need to store a different value than default numeric values 0 and 1, for example a flag that is stored with N and Y. Here's a code snippet of a check-box with a pic x variable set to N or Y:

```
77 w-flag pic x.
...
03 check-box ...
  check-off-value "N" check-on-value "Y"
  value w-flag.
```

This new syntax is also useful when placing the check-box controls inside a grid using the statement "Display Check-Box upon GridName (-1, 3)" to show different values in the grid filter when the column includes a check-box.

- Radio-Button supports alphanumeric values in the property Group-Value, allowing a pic x variable to be used to hold the selection in a radio button group instead of the default numeric variable. Here's a code snippet of radio buttons with a pic x(n) variable than can be set to "A" "B" or "C":

```
77 w-type pic x.  
...  
03 radio-button ...  
  group 1 group-value "A" value w-type.  
03 radio-button ...  
  group 1 group-value "B" value w-type.  
03 radio-button ...  
  group 1 group-value "C" value w-type.
```

- Push-Button supports a new property named ROLLOVER-BORDER-COLOR to specify the border color of flat buttons when the mouse hovers on the control and can be used in conjunction with the existing properties Rollover-Background-Color and Rollover-Foreground-Color. A code snippet of a button with the new property is shown below:

```
03 push-button flat ...  
  rollover-background-color rgb x#A6F9DE  
  rollover-foreground-color rgb x#D51515  
  rollover-border-color   rgb x#990066.
```

- Grid and List-Box controls support a new property named EXPORT-FILE-OPEN to automatically open an Excel file after it has been produced by the Export feature. This simplifies the opening of the files created on the client. Here's a code snippet of controls with the new property:

```
03 list-box ...  
  export-file-name w-path  
  export-file-format "xlsx"  
  export-file-open 1.  
03 grid ...  
  export-file-name w-path  
  export-file-format "xlsx"  
  export-file-open 1.
```

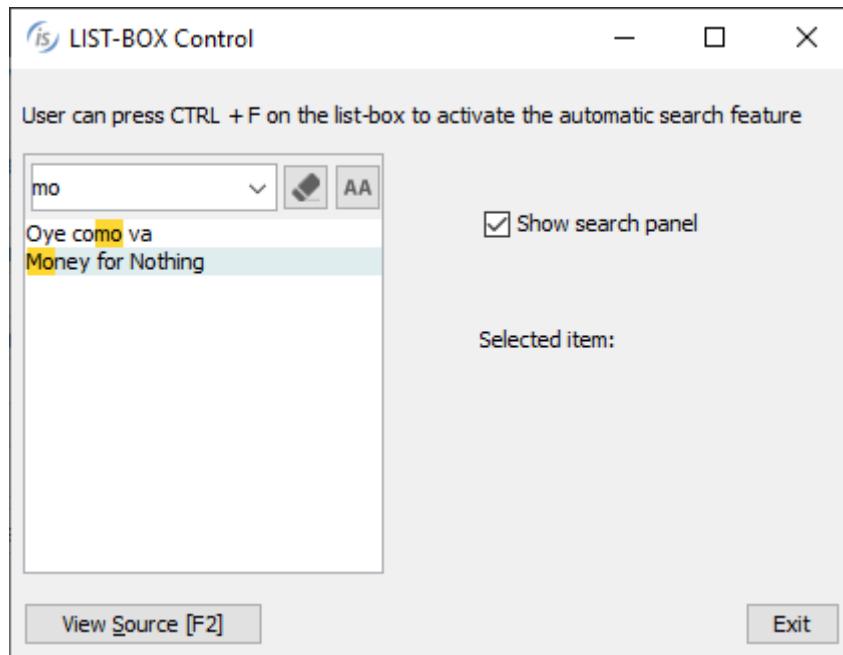
- List-Box now supports the SEARCH-PANEL property using the same rules as defined in the grid control to manage the search feature. By default, this property is set to 0, making the search panel appear on top of the control only when the user presses the Ctrl-F keyboard shortcut. By setting the property to 1, the search panel will always be visible at the top of the List-box. To completely disable the search panel, the property can be set to -1. You can change the shortcut key used to open the search panel with the keystroke configuration setting. To set the shortcut to Ctrl+G instead of Ctrl+F, for example, use the setting:

```
iscobol.key.*g=search=list-box
```

The following is a code snippet of a list-box with the new property set to 1 and Figure 4, *List-box search-panel in action*, showing the search panel used to filter the content of items loaded in the list-box.

```
03 list-box ...
  search-panel 1
```

Figure 4. List-box search-panel in action.



- Tree-View supports a new MSG-MOUSE-CLICKED event, fired by both the standard Tree and Table-View tree when the NOTIFY-.MOUSE style is set. This allows an action to be performed when clicking on an item instead of the standard double-click. When used on a Table-View, the special name EVENT-DATA-1 is set to the column number where the click happened. Here's a code snippet to use the new event supported:

```
03 tv1 tree-view table-view ...
  display-columns (1, 20)
  notify-mouse event tv1-event.
...
tv1-event.
  Evaluate event-type
  ...
  when msg-mouse-clicked
    if event-data-1 = 1 | click on first column
  ...
  end-if
end-evaluate.
```

## isCOBOL Runtime

Improvements to library routines and configuration settings have been added to the new isCOBOL runtime:

## Improvements to library routines

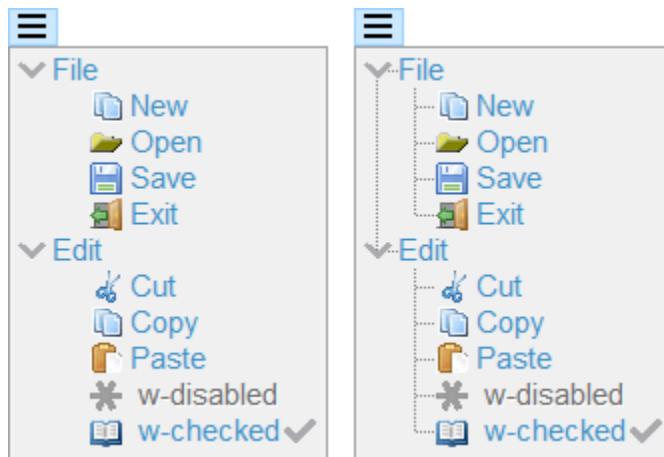
Several library routines have been improved in the 2022 R2 release:

- W\$MENU library supports a new attribute named "show-lines" to show the lines connecting menu items in the hamburger menu. The following code snippet activates the new attribute:

```
call "W$MENU" using wmenu-set-attribute "show-lines" "yes"
```

The result when running the snippet is shown in Figure 5, *Show-lines attribute of Hamburger menu*. The menu on the right has the new attribute set to "yes".

Figure 5. Show-Lines attribute of Hamburger menu.



- C\$COPY and C\$FSCOPY routines support an additional optional parameter to specify that the indexed file to be copied is encrypted. This works in conjunction with the JISAM file handler by using the same key set in the configuration option *iscobol.file.encryption.key*. The code snippet below shows the usage of the new parameter:

```
call "C$COPY" using src-path, dest-path, "I", 1
call "C$FSCOPY" using src-path, dest-path, 1
```

- WIN\$PRINTER library is now more flexible when creating PDF files using the -P PDF flag or saving a PDF file in the Print Preview. With previous releases it was mandatory to CALL the WINPRINT-SET-ATTRIBUTE op-code to set the attribute FONT\_FOLDER or FONT\_FOLDER\_EMBED to embed fonts in the PDF file. In the current release, if these attributes are not set, isCOBOL will search for fonts in the default operating system's font folder: C:\Windows\Fonts on Windows, /usr/share/fonts on Linux and /Library/Fonts on MacOS, to embed fonts and use the Identity-H encoding in the generated PDF file.

## New configurations

New configuration settings have been added in this release:

- *iscobol.file.index.open\_hook=ProgramName* can be used to specify a hook program to be called before a file is opened, and allows the file path or file handler class to be modified on the fly. The ProgramName must declare two parameters received in linkage section as follows:

```
linkage section.  
77 file-path pic x(300).  
77 file-class pic x(300).  
procedure division using file-path, file-class.
```

The hook program is executed just before the file is opened and, if necessary, can update the value of the two parameters to change the file path or the file handler. This flexible solution can be used, for example, to easily migrate programs from a Fat-Client to a Thin-Client architecture, where the “local working files” are not on the client PC, but are located on the server, and files might have to be kept separate for different users.

- *iscobol.floating\_point\_format=ibm\_hfp* to store data-items with usage ‘float’ or ‘double’ using the IBM hexadecimal floating-point format (known also as HFP by IBM), instead of the default “ieee\_754”, the technical standard for floating-point arithmetic established in 1985 by the Institute of Electrical and Electronics Engineers (IEEE). In comparison to IEEE 754 floating point, the HFP by IBM format has a longer coefficient, and a shorter exponent.
- *iscobol.gui.nested\_embedded\_proc\_check=true* to check for nested accepts in embedded procedures. When this configuration option is set, a specific exception is thrown. This helps the developer easily identify and correct any nested accepts found in the code.
- *iscobol.file.index.ctshmemdir=directory* to set the sharedmem directory for the CTREEJ file handler. On Unix platforms the c-treeRTG server shares a directory with the c-treeRTG clients when the communication is performed using the shared memory protocol. When using multiple c-treeRTG servers, it’s important to configure the SHMEM\_DIRECTORY in each server’s ctsrvr.cfg configuration files, and set the same value in this new isCOBOL configuration setting, so that every server uses a separate folder for SHAREMEM communication.
- *iscobol.file.index.endiancheck=false* to skip the byte endianness check made by CTREEJ file handler during connection. By default, the check is made to ensure that the byte endianness of both client and server is the same. In cases where this check is not necessary, such as when accessing files not containing native numeric data types, the check can be relaxed so the connection can succeed.

Additionally, it’s now possible to configure the exception values returned when the user just presses a letter without any additional special key. This helps in situations such as when accepting a screen containing only controls not used for editing, for example pushbuttons, allowing the user to quickly choose an option by just pressing a letter without the need to press it in combination with Alt or Ctrl. For example, configuring:

```
iscobol.key.a=exception=101  
iscobol.key.b=exception=102
```

when the user presses the “a” key, the program intercepts the exception-value 101, and 102 is received when pressing the “b” key.

## isCOBOL Debugger

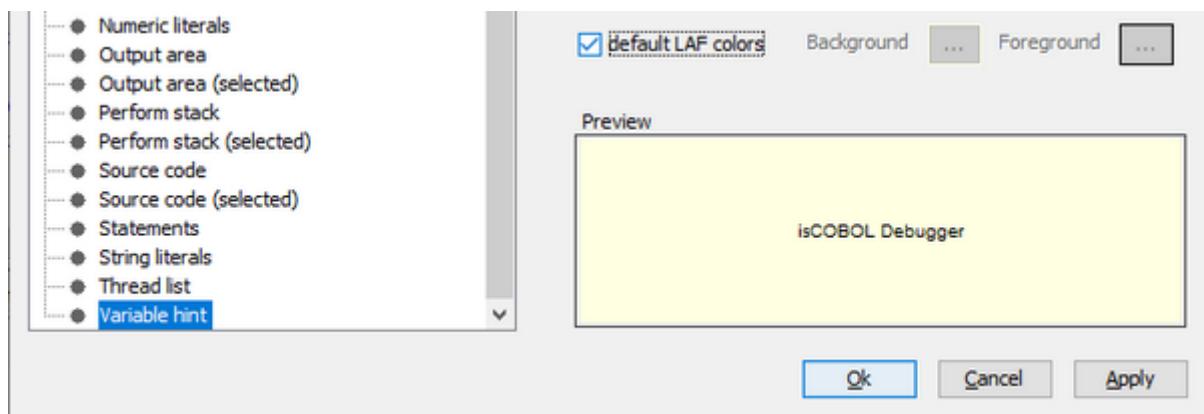
The isCOBOL Debugger has been enhanced in the 2022 R2 release to improve the usability when using different LAFs. These are the features added in the new release:

- colors configured in Debugger can now follow the standard LAF color settings

The debugger's "Fonts and Colors" window, from the menu's Settings/Customize option, is where you would set the different fonts and colors for different parts of the Debugger.

In 2022 Release 2, you can set the "default LAF colors", and the result will be different when running on different operating systems or when running with a different LAF. The colors will then be more coherent, for example the Hint displayed when the mouse is on a data item, will look the same as the Hint displayed for the buttons. The new setting is shown in Figure 6, *Debugger LAF color*.

Figure 6. Debugger LAF color.



- new option -a for Debugger's infostack command to show all available info

The *infostack* is a command that can be used in the command-line to list all paragraphs and programs involved in the stack. With the new option –a, the source file and line number are shown in the output. For example this is the output of the command with the new –a option executed while debugging:

```
+ MAIN [THREAD_BRIDGE] source/THREAD-BRIDGE.cbl:34
+ MAIN [ISCUSTOMER] source/ISCUSTOMER.cbl:265
+ AFTER-ACCEPT [ISCUSTOMER] source/ISCUSTOMER.cbl:291
+ READ-NEXT [ISCUSTOMER] source/ISCUSTOMER.cbl:424
```

## WebClient

isCOBOL WebClient 2022 R2 is Veryant's solution for running desktop applications in a browser, and it has been updated to support Java 17, to complete the process started with the 2022 R1 release to certify all Veryant products with all current LTS Java versions: 1.8, 11 and 17.

The new version of WebClient has been optimized to better handle low speed and high latency connections. It now also supports window transparency, and has improved touch screen integration. The session recording and mirroring and new Admin console are among the many other improvements.

## isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. As of version isCOBOL 2022 R2, the `HTTPClient` class can consume HEAD requests to retrieve only header fields.

### HTTPClient

`HTTPClient` is a class that enables COBOL programs to interact with Web Services. This class has been updated with these new method signatures:

```
public void doHead (String strUrl, HTTPData.Params p)
public void doHead (String strUrl)
```

The new methods will perform an HTTP HEAD request on the provided URL and using optional HTTP parameters.

The HTTP HEAD method requests HTTP headers from the server as if the document was requested using the HTTP GET method. The only difference between HTTP HEAD and GET requests is that for HTTP HEAD, the server only returns headers without the body.

The HTTP HEAD method is much faster than the HTTP GET method because much less data is transferred in HEAD requests. Browsers use the HEAD method to update information about cached resources to check if the resource has been modified since the last time it was accessed. If the resource has not been modified, browsers reuse the local copy without issuing a new request. Otherwise, they request an updated version of the resource with a GET request.

The HEAD method can be also used to check if a file on the server exists, without actually downloading it if not really necessary.

## Additional improvements

isCOBOL 2022 R2 release improves the JUTIL utility.

A new option `-make` is now supported to create an empty Jisam file from the xml dictionary. This approach is equivalent to the `ctutil -make` option already supported by `ctutil` for c-TreeRTG.

Here's an example of the new option used to create an empty Jisam file named "products" based on its definition stored in "product.xml":

```
jutil -make products products.xml
```

Note that "products.xml" is the file dictionary that describes the indexed file with all its details and is created by isCOBOL's compiler when using the `-efd` option.

## isCOBOL Evolve

---

# isCOBOL 2022 Release 1 Overview

## Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2022 R1.

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

The new version now supports all Java LTS releases: Java 1.8, Java 11 and Java 17.

isCOBOL IDE is now based on a newer Eclipse version, 2021-09 (4.21).

WebClient now supports balancing through clustering and a restyled administration UI.

2022R1 also includes many GUI improvements and new compatibility features to simplify the migration of existing COBOL compiler to isCOBOL Evolve suite.

Details on these enhancements and updates are included below.

## isCOBOL Runtime

isCOBOL Evolve 2022R1 is certified for Java 17, the current Java LTS (Long Term Support) release. It is important to have isCOBOL Evolve 2022R1 qualified to be used by the latest LTS version of Java because LTS applies the tenets of reliability engineering to the software development process and software release life cycle. Long-term support extends the period of software maintenance; it also alters the type and frequency of software updates (patches) to reduce the risk, expense, and disruption of software deployment, while promoting the dependability of the software. It does not necessarily imply technical support. In addition, starting from Java 17, Oracle JDK 17 and future JDK releases are provided under a free-to-use license until a full year after the next LTS release.

isCOBOL Evolve 2022R1 now supports all Java LTS releases: Java 1.8, Java 11 and Java 17. WebClient currently does not support Java 17, but we plan to have it ported in the 2022R2 release.

In addition to Java 17, isCOBOL Evolve 2022R1 is also fully qualified to be used under Microsoft Windows 11 in combination with Java 11.0.13 (see <https://www.oracle.com/java/technologies/javase/products-doc-jdk11certconfig.html>)

New library routines and configuration settings have been added to allow multiple configuration files to be used when running isCOBOL applications.

## New configurations

New configuration settings have been added in this release:

- *iscobol.conf.copy=FilePath* to include a separate configuration file in the current configuration file. This setting can be used to easily maintain and use multiple configuration files. For example, when running the following command:

```
iscrun -c myapp.properties MAINPROG
```

and the myapp.properties file contains:

```
iscobol.conf.copy=default.properties  
...  
iscobol.conf.copy=customer-specific.properties
```

the program is run using the settings specified in the myapp.properties configuration file, along with settings from the default.properties and customer-specific.properties files. If the same configuration property name is set in multiple configuration files, the last defined one takes precedence over the others, thus overwriting previous values.

- *iscobol.file.index.fileversion=n* to create c-tree files that are backward compatible. For example, you can set the value to 2 to make the current C-Tree RTG v3 create files compatible with the previous C-Tree RTG v2.
  - *iscobol.file.output\_lock=false* to prevent an exclusive lock during open output. The default value is True, so that an OPEN OUTPUT is treated as OPEN OUTPUT WITH LOCK.
  - *iscobol.file.extend\_lock=false* to prevent an exclusive lock during open extend. The default value is True, so that an OPEN EXTEND is treated as OPEN EXTEND WITH LOCK.
- The previous two configurations can be set in rare cases of lock needs. By default, they are set to true for higher performance of WRITE operations after the file is opened.
- *iscobol.display\_message\_timeout=n* to specify a timeout in hundreds of a second for error message boxes. When the timeout expires, the error message box is automatically closed as if the user pressed the OK button. When set in the runtime configuration it affects every Java exception that is usually reported via graphical message box. When set in the isCOBOL Client local configuration it affects connection error and session termination messages.

Additionally, the existing configuration *iscobol.file.page\_eject\_on\_close=true* is now supported on files assigned to -P SPOOLER-DIRECT as well as other print files.

## New library routines

Two new library routines have been implemented:

- C\$CONFIG to reset, load or append a new configuration properties file. This library routine can be used to reset the runtime configuration by setting a new configuration file, or to append a configuration file to the current configuration after the main program is started. It can be useful, for example, to load a different configuration file after logging in. The following snippet:

```
call "c$config" using cconfig-reset,  
      "conf2.properties"
```

resets the currently loaded configuration and loads a new configuration contained in the conf2.properties file.

This code snippet:

```
call "c$config" using cconfig-append,  
      "conf3.properties"
```

appends settings contained in the "conf3.properties" file to the current configuration.

- W\$GETC is used to retrieve the keystroke pressed. This routine is useful when the program needs to perform a low-level intercept of the key pressed in a character-based user interface.

The following code snippet shows how to intercept the F1 key:

```
77 wcharacter          pic xx.  
...  
call "w$getc" using wcharacter  
if wcharacter = "k1"  
  display "pressed F1"
```

These routines can be used in new programs, and also increase compatibility with other COBOL dialects, such as ACUCOBOL-GT.

A new op-code named WINPRINT-GET-NO-ASYNC-JOBS has been added to the WIN\$PRINTER routine to inquire how many async print jobs are currently running. This is useful when using asynchronous print jobs to check if some are still running. For example, this code snippet checks if the async jobs previously executed are terminated:

```
call "win$printer" using WINPRINT-GET-NO-ASYNC-JOBS, n-jobs  
      giving winprint-status  
if n-jobs = 0  
  ...
```

## isCOBOL IDE enhancements

The isCOBOL IDE 2022 R1, now based on Eclipse 2021-09 (4.21), requires at least Java version 11 by default, but offers full support for Java 17. There are many improvements in performance stability and usability on a day-by-day usage. Below is an extract of the new functionalities:

- A new "System" theme is available in the Appearance preference page. This theme is built using system colors, and as a consequence integrates well with any OS and OS theme. Also, there were many enhancements on "dark" theme to style better menus, drop down controls, tool-tips, progress bars and more.
- A new preference, Enable word wrap, is available in the Console preference page. This setting persists the current state of the "Word wrap" toggle onto the console view between user sessions. By default, word wrapping is disabled on console output.
- In the Console view, you can repeat your last search in the forward or backward direction in the following ways: Right-click in the Console view, then select Find Next or Find Previous.
- Previous Edit Location navigation (formerly named Last Edit Location) is now expanded to remember multiple edit locations. The last 15 edit locations are now remembered. For convenience, similar edit locations in close proximity to each other are also merged so that each of

the 15 remembered locations remains distinct. Ctrl+Alt+LEFT\_ARROW (or on Mac Ctrl+Opt+LEFT\_ARROW) navigates to the most recent edit location, just as Ctrl+Q always has in prior releases.

- You can now scroll horizontally in the Text Editor using Shift+Mouse Wheel and touchpad gestures on Windows. Horizontal scrolling with touchpad already works on Linux and MacOS.

## WebClient

isCOBOL WebClient, Veryant's solution for running desktop applications in a browser, has a more modern and better organized administration console in the 2022R1 release that makes configuration easier, and includes a new balancing solution.

### New administration user interface

The admin user interface has received an overhaul, and is now more structured in how all parameters are presented to the administrator. It provides categories that logically group parameters, making it much easier to navigate settings, both for server configuration and application configuration.

A server status page lets you keep track of the number of session pools available to applications, the number of currently active users, the number of running sessions, how many connections are established, and the number of configured and enabled applications.

The sessions' view allows administrators to monitor currently running applications, and, if allowed, to view and take control of users' sessions.

Figures 1, 2 and 3 depict the new administration panel, and the new grouping of the configuration items in categories.

Figure 1 – Server configuration page

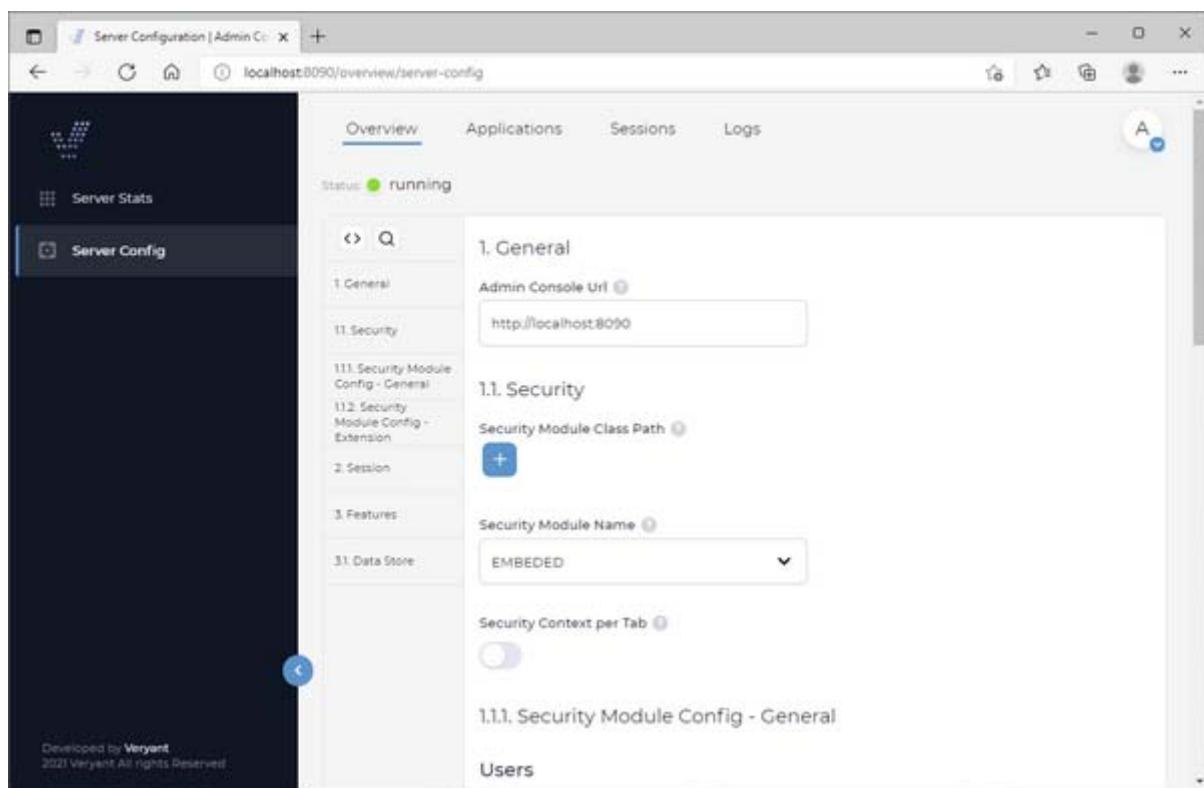


Figure 2 – Web application configuration

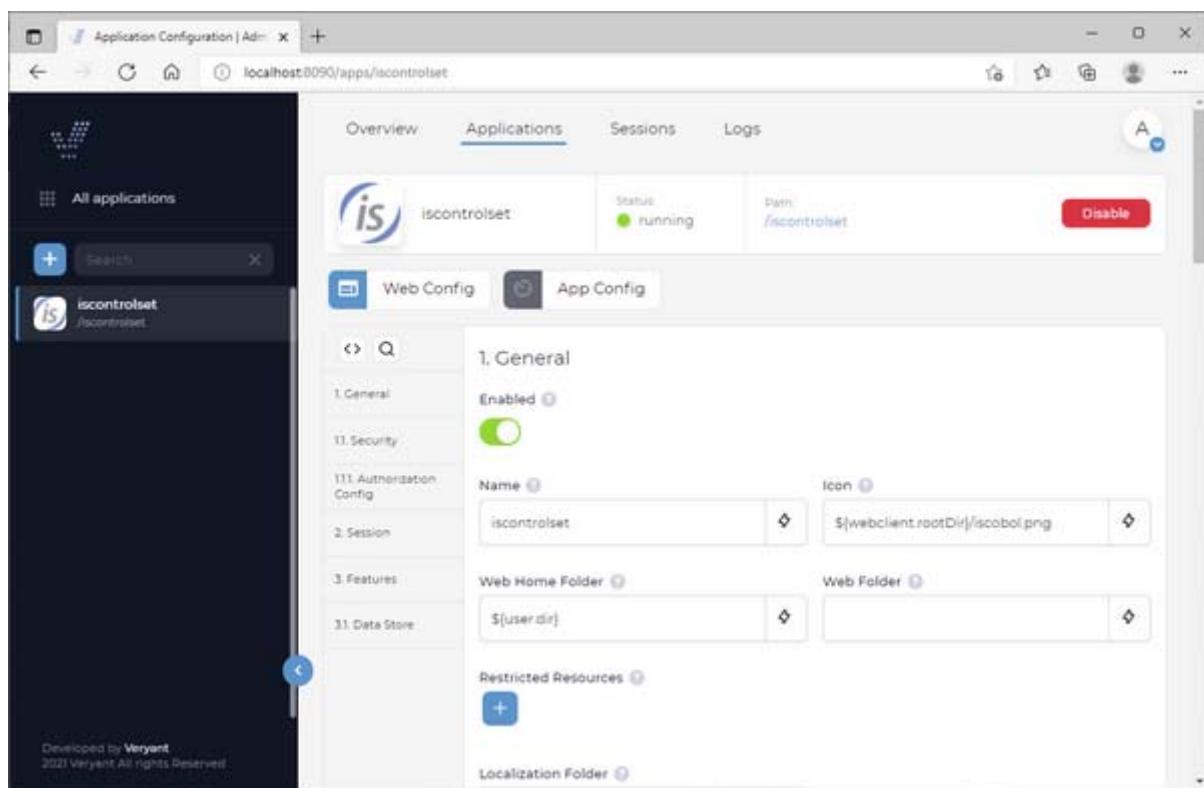
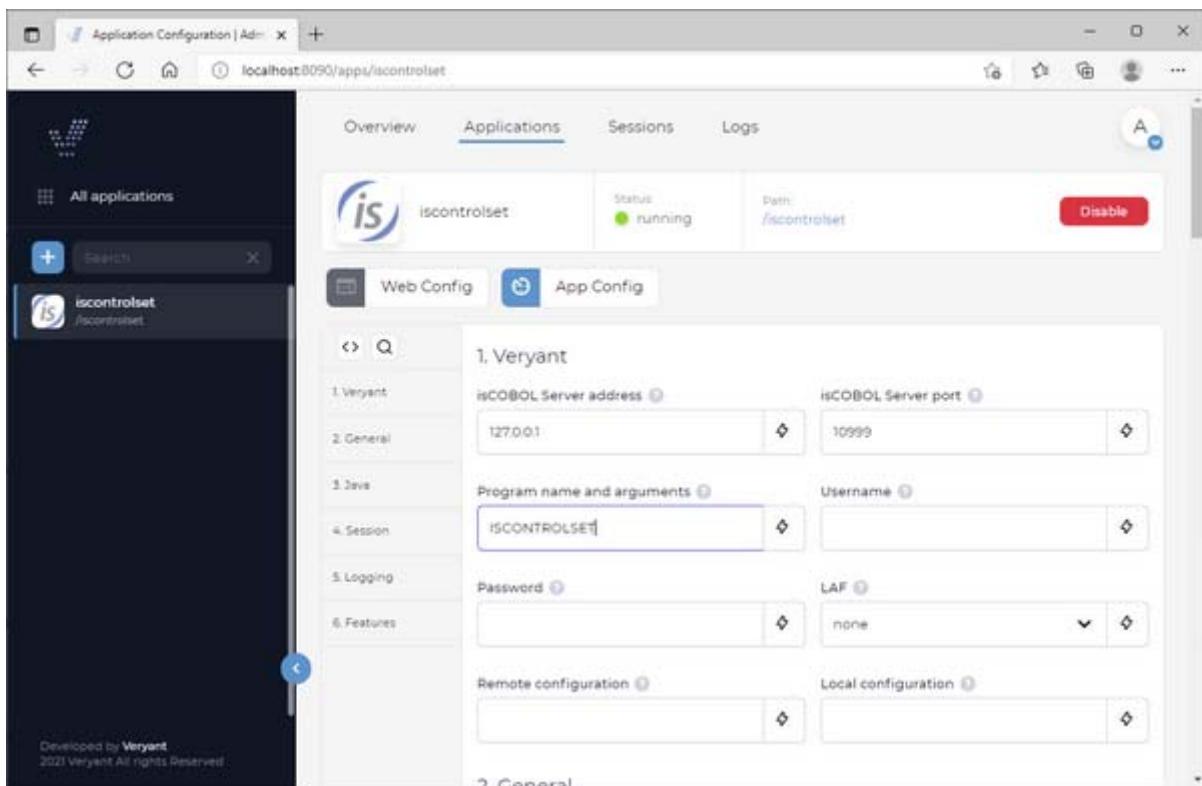


Figure 3 – Application configuration



## WebClient Balancer

WebClient now supports balancing through clustering; separating WebClient into modules that can be divided into multiple instances on multiple servers to enable you to handle larger loads. These modules are

- WebClient Cluster Server which handles web requests
- Application pool which handles running COBOL applications
- Admin Console which handles the server administration pages.

A minimal installation requires 1 cluster server and 1 session pool, and more cluster servers can be added for redundancy. As a general rule, no more than 2 cluster servers should ever be needed in most cases, as it is a very lightweight task that basically resends messages from the application instance and the browser. It also picks an application pool in which to run the requested application.

Session pools can be scaled dynamically, based on the number of connected users or resource usage. Cluster Server is a stateless web server, which means that if you have multiple Cluster Servers it doesn't matter which server you connect to from the browser, even on refresh, the browser connection will always find the way to its application instance. If you have multiple Cluster Servers deployed and one of the Cluster Servers terminates, all the browser connections to this server can be reconnected through the other running Cluster Server without terminating the application instance.

Inside Cluster Server is a built-in mechanism - a Session Pool balancer. This mechanism is responsible for finding a free Session Pool where a new instance will be started. The WebClient balancer uses a round-robin algorithm.

Everything in your application that is related to web is configured and provided through the Cluster Server. This includes security (SSL, authentication, and logout), translations, fonts and custom web resource handling. On the other hand, everything related to your application binary is configured and handled in Session Pool. Because of this the `webclient.config` file needs to be divided into 2 separate configuration files - `webclient-server.config` which is used by Cluster Server and `webclient-app.config` which is used by Session Pool.

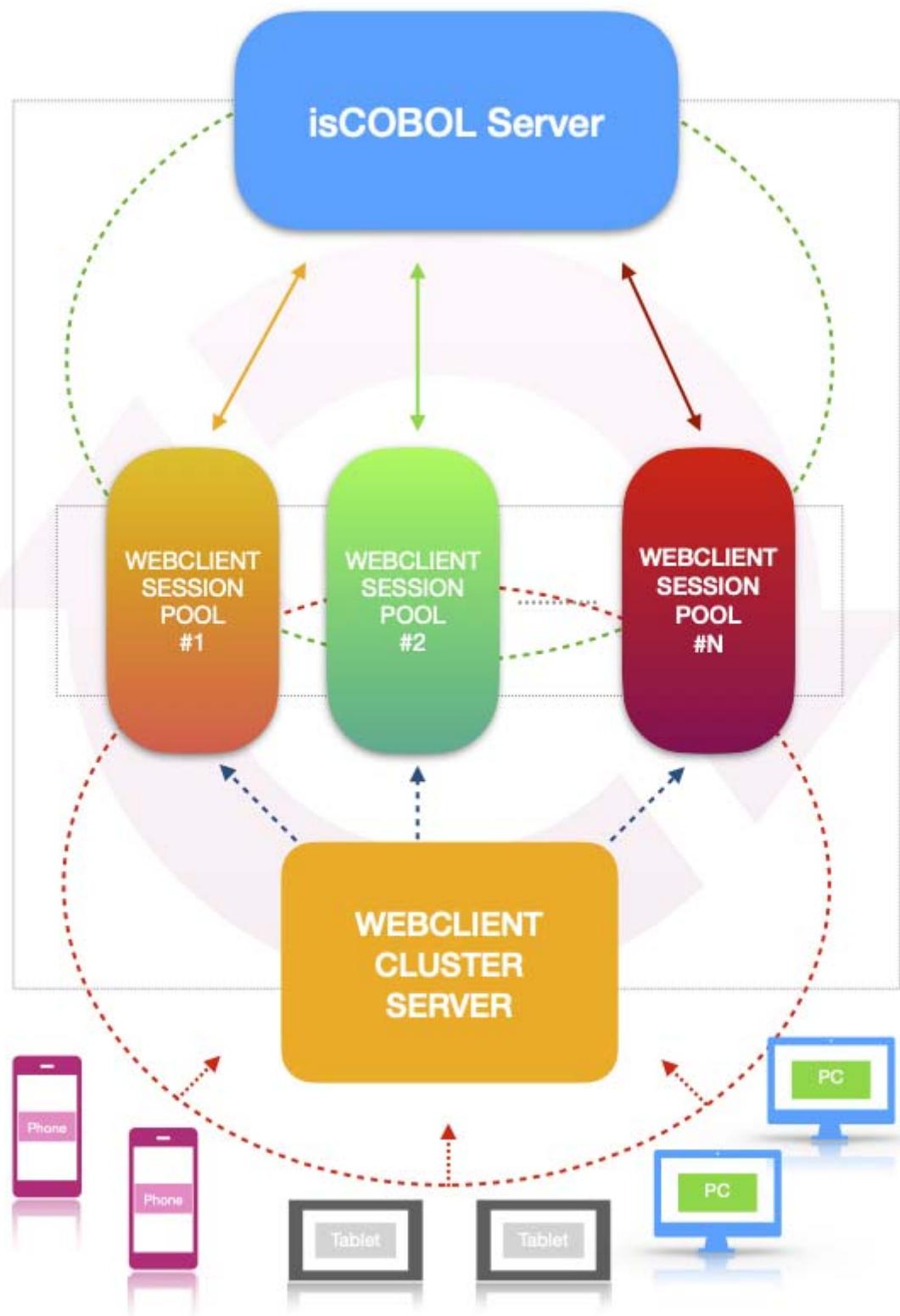
A single Admin console can be used to handle configuration of all Cluster Servers by configuring the list of websocket URLs of all Cluster Servers in the Admin Console configuration file.

Modules inside a Cluster deployment are connected through WebSocket. To make sure that no other malicious servers connect to your Cluster Servers, all modules must share the same secret key which is used in handshake when establishing the websocket communication. The secret key is defined in `webclient.properties`, `webclient-sessionpool.properties` and `webclient-admin.properties` as property `webclient.connection.secret`. Please change this secret key to your random value before going into production. This secret key is also used for signing JWT tokens that are used for user session cookies. Make sure you use only alphanumeric characters in the secret key, and do not use any special characters.

Figure 4 – *Communication flow* describes the flow of communication between devices running the application. A device connects to the WebClient balancer, which connects to a Cluster Server. The server will run an application using one of the available session pools.

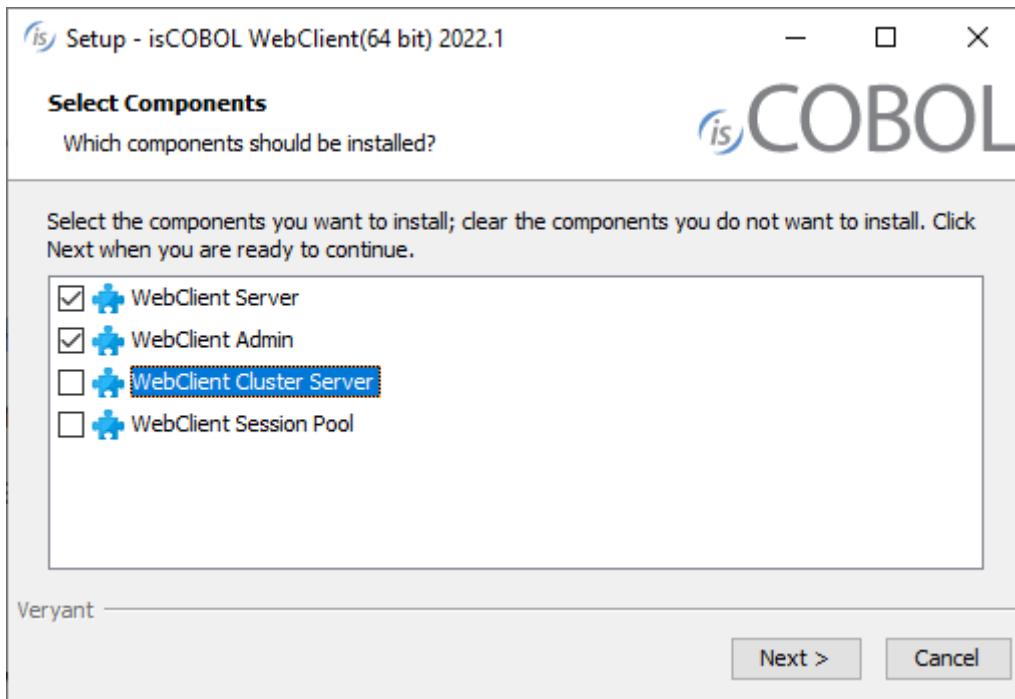
Figure 4 – Communication flow





To work with the new cluster solution, we removed WebClient product from the isCOBOL SDK installer and we created a new installer for WebClient. In the new installer, named WEBC, you to choose which components you need to install as shown in Figure 5, *New WEBC installer*.

Figure 5. New WEBC installer



The WebClient Server and WebClient Admin components are the standard ones, already available in previous releases. These products start processes using the wrapper commands `webclient` and `webclient-admin`, or the equivalent `webclient` and `webclient-admin` for services.

The new component named WebClient Cluster Server can be started with the new wrapper command `webclient-cluster` or `webclient-cluster` for services.

The last component, named WebClient Session Pool, can be started with the new wrapper command `webclient-session` or `webclient-session` for services.

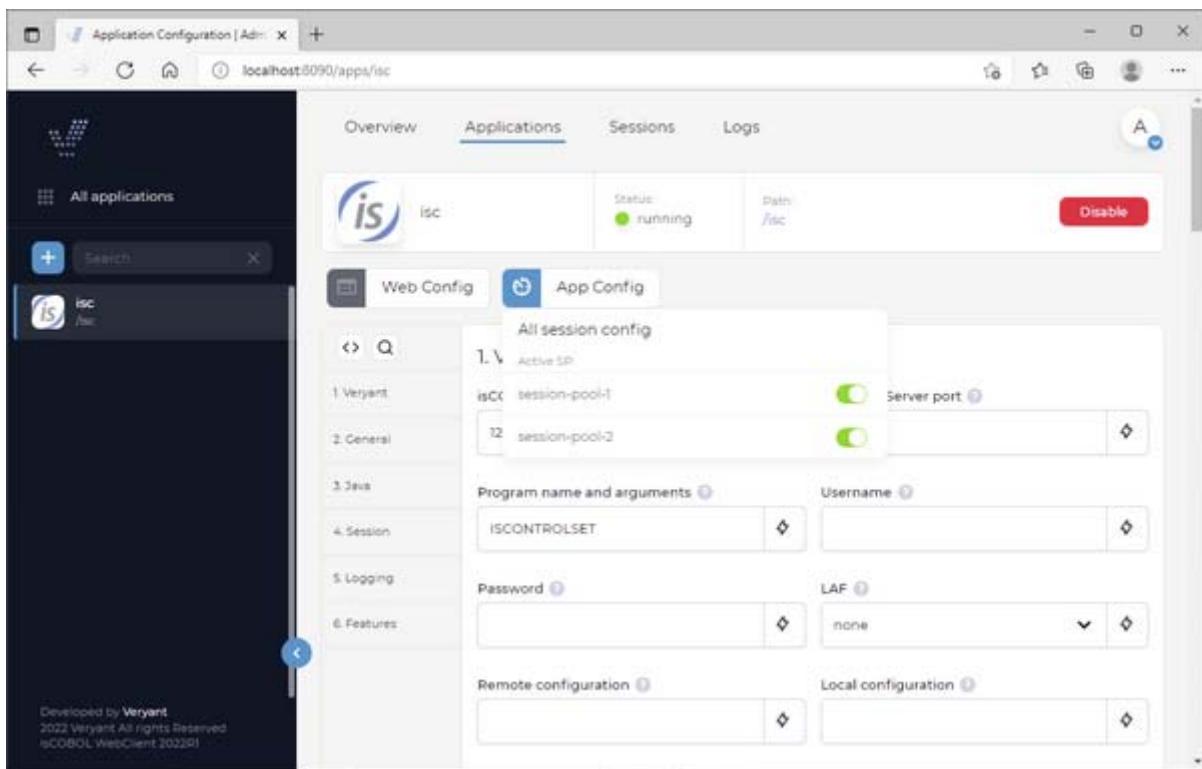
After configuring and starting the appropriate processes on the different servers, the WebClient Admin web application will look like in Figure 6, *Session Pools shown in the Admin page*.

Figure 6. Session Pools shown in the Admin page

The screenshot shows the 'Overview' tab in the Admin Console. On the left, there's a sidebar with 'Server Config' and 'Session Pools' sections. The main area has tabs for 'Overview', 'Applications', 'Sessions', and 'Logs'. Under 'Overview', it shows 'Cluster servers' with one entry: 'Webclient Server' (localhost:8080) with 2 session pools, 0 active users, 0/0 sessions, 0 connections, and 1/1 app running. Below that is a 'Session Pools' section with two entries: 'session-pool-1' and 'session-pool-2', both of which are running with 0 sessions, max sessions set to Unlimited, priority 1, and 2/1 applications.

So, from the Applications tab you can associate the desired Session Pools through the clock icon in the App Config button, as shown in Figure 7, *Associate Pools to the App*.

Figure 7. Associate Pools to the App



In addition, starting from 2022R1 release a new wrapper command is available to start in just 1 process both the WebClient Server and the Admin. The name of the wrapper is `webclient-and-admin` or `webclient-and-admin` for services. This approach, simplify the testing environment, but is not suggested for production environments.

## GUI enhancements

Many improvements to GUI controls have been implemented in this release. Library routines `W$BITMAP` and `W$MENU` now allow merging different symbol fonts and to set new attributes in the Hamburger menu. New configuration settings can be used to modify the look and feel of modal windows and to have finer control with the `LM_ZOOM` layout manager.

### GUI controls

A new property named `MARGIN-WIDTH` is supported in Entry-Field controls to set or retrieve the spacing between text and borders. The property is a list of four values that specify the padding space in pixels from the top, left, bottom and right border respectively (the same rule of the existing `BORDER-WIDTH` property). When used on single line entry-fields, only the left and right borders are meaningful, since text is always vertically centered.

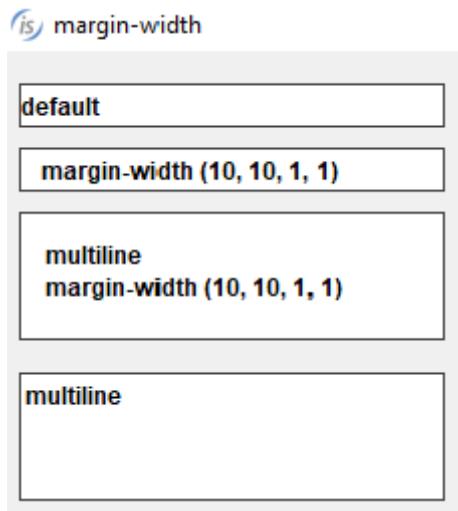
The same spacing applies to the `PLACEHOLDER` property, if defined. This feature, known also as Inset, helps in defining different spacing of different controls. Alternatively, you can use a different margin space in all entry-field controls of the entire isCOBOL application by using the compiler code injection feature.

For example, the entry-fields defined below use the new syntax in the second and third entry-fields:

```
03 entry-field  
    line 2 col 2 size 30 cells.  
03 entry-field  
    line 4 col 2 size 30 cells  
    margin-width (10, 10, 1, 1).  
03 entry-field multiline line 6 col 2 lines 4 cells size 30 cells  
    margin-width (10, 10, 1, 1).  
03 entry-field multiline line 11 col 2 lines 4 cells size 30 cells.
```

The result when running is shown in in Figure 8, *Entry-field MARGIN-WIDTH property*.

Figure 8. Entry-field MARGIN-WIDTH property



The FLAT style is now supported in Tree-View controls to have more modern-looking buttons when running with the Windows LookAndFeel. For example, having this code in the screen section declaration:

```
01 Mask.  
03 Tv1 tree-view buttons lines-at-root  
    line 2 col 2 lines 17 size 40 cells  
    show-sel-always selection-background-color rgb x#6883AE  
        selection-foreground-color rgb x#FFFFFF.  
03 Tv2 tree-view buttons lines-at-root FLAT  
    line 2 col 44 lines 17 size 40 cells  
    show-sel-always selection-background-color rgb x#6883AE  
        selection-foreground-color rgb x#FFFFFF.
```

with the FLAT style is set only in the second tree-view, will result in the running program looking as shown in Figure 9, *Tree-view FLAT style*, where you can notice the difference in the styling of the two tree view controls.

Figure 9. Tree-view FLAT style



The TRANSPARENT-COLOR property, already supported with BITMAP controls, is now supported on PUSH-BUTTON controls. A specific color can be specified to represent the transparent color in the button's bitmap image, even if the bitmap has no transparency.

The layout-manager syntax now supports "max-font-zoom=n min-font-zoom=n" clauses to set the maximum and minimum zoom levels that can be applied during window resizing when used in conjunction with the LM-ZOOM layout manager. The values are expressed as a percentage in which the font will be altered. When the zoom level is outside this range, LM-ZOOM behaves exactly as LM-SCALE, resulting in changes only to the dimensions and coordinates of the controls without affecting the font size. For example, the code snippet:

```
77 my-lm-zoom handle of layout-manager LM-ZOOM
      "max-font-zoom=150 min-font-zoom=50".
```

declares a layout-manager for LM-ZOOM allowing font changes from 50% when decreasing and 150% when increasing.

A new event named MSG-FINISH-FILTER is fired in grid controls when the visible rows in the grids change because of changes applied using the Filters or Search panel features. This is useful if the program needs to be aware of a filter being applied to the grid's content.

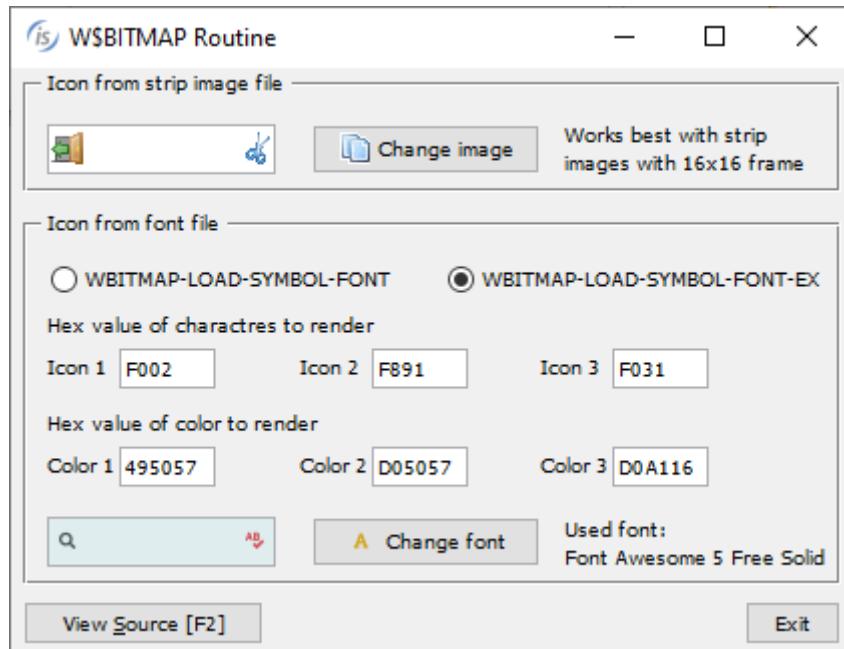
## Library routines

A new op-code named WBITMAP-LOAD-SYMBOL-FONT-EX in W\$BITMAP library has been implemented to load icons and to merge different font symbols or font styles. This lets you create a handle with a strip of icons loaded from different font handles, or from the same handle but with different styles, such as the color. The code snippet:

```
initialize wbitmap-lsf-data
move h-font      to wbitmap-lsf-font(1)
move character-1-n to wbitmap-lsf-characters(1)
move color-1-hex  to wrk-color-hex
perform CALCULATE-COLOR
move wrk-color    to wbitmap-lsf-color(1)
move h-font      to wbitmap-lsf-font(2)
move character-2-n to wbitmap-lsf-characters(2)
move color-2-hex  to wrk-color-hex
perform CALCULATE-COLOR
move wrk-color    to wbitmap-lsf-color(2)
move h-font      to wbitmap-lsf-font(3)
move character-3-n to wbitmap-lsf-characters(3)
move color-3-hex  to wrk-color-hex
perform CALCULATE-COLOR
move wrk-color    to wbitmap-lsf-color(3)
call "W$BITMAP" using wbitmap-load-symbol-font-ex,
           16, wbitmap-lsf-data
           giving h-font-icon
```

creates the handle h-font-icon with 3 icons from different characters of the same font handle (h-font in this case loaded from Font Awesome 5 Free Solid) but using 3 different colors, and the 3 images can be used with the standard bitmap-handle and bitmap-number syntax as shown in the entry-field and push-button in Figure 10, *WBITMAP-LOAD-SYMBOL-FONT-EX new op-code*.

Figure 10. WBITMAP-LOAD-SYMBOL-FONT-EX new op-code



New attributes for the W\$MENU library have been added to customize the hamburger menu layout and behavior. This is the list of new attributes:

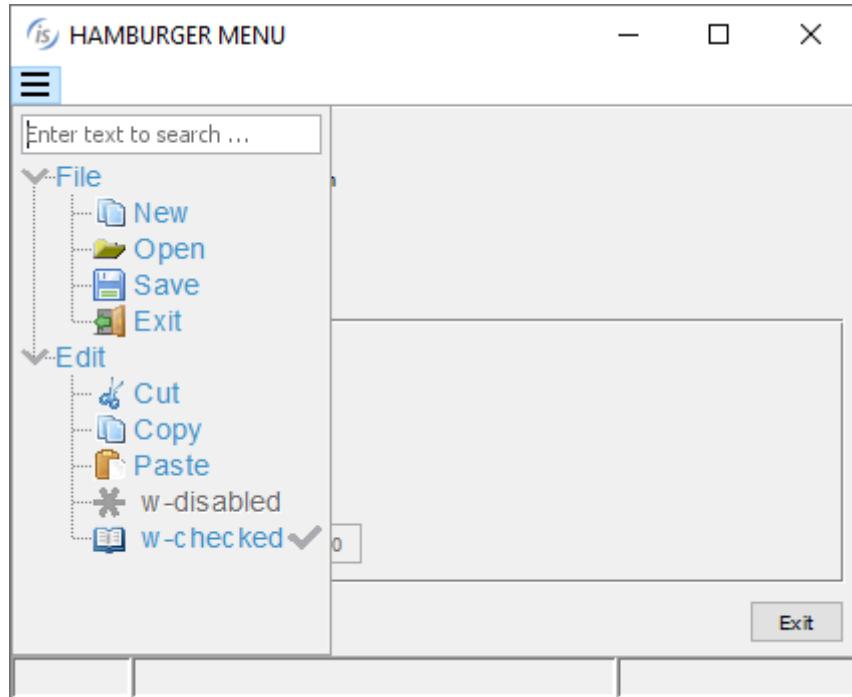
- EXPANDED, to specify if the menu items with children should be automatically expanded
- HEIGHT, to set the height in pixels of the area covered by the hamburger menu
- LAYOUT-MANAGER, to set the rules of layout-manager bound to the menu that will applied when the window is resized while the menu is open
- SEARCH-PANEL, to enable the tree view's search panel, allowing the user to filter the content
- STATUS-BAR-COVERING, to prevent the hamburger menu's tree view to overlap the window's status bar.

For example, this code snippet sets three of the new attributes:

```
call "w$menu" using wmenu-set-attribute, "expanded", "yes"
call "w$menu" using wmenu-set-attribute, "search-panel", "yes"
call "w$menu" using wmenu-set-attribute, "status-bar-covering", "no"
```

and in Figure 11, *Hamburger menu new attributes*, you can see the new look of the hamburger menu when opened.

Figure 11. Hamburger menu new attributes



The W\$CREATEFONT library routine can now load fonts that have been embedded in the class using the COPY RESOURCE directive. This allows you to use a font without needing it to be installed on the client PC. For example, this code snippet:

```
copy resource ".../resources/Font Awesome 5 Free-Solid-900.otf".
call "w$createfont" using "Font Awesome 5 Free-Solid-900.otf"
          "Font Awesome 5 Free Solid".
```

uses the COPY RESOURCE directive to include that .otf file in the class at compile time, and the W\$CREATEFONT to load it.

## Configuration settings

New configuration settings have been implemented:

- *iscobol.gui.windows\_darkening=n* to specify the opacity of the parent window when modal floating windows and message boxes are opened. A positive value ranging from 1 to 100 specifies the level of darkening with 1 representing an imperceptible darkening and 100 representing the black color, while a negative value ranging from -1 to -100 specifies the level of transparency with -1 representing an imperceptible transparency and -100 representing full transparency.

This is especially useful with applications running in WebClient, to emulate the effect of modern web applications when modal windows are opened. Thin Client applications can also take advantage of this property for a more modern look and feel of the application.

For example, configuring: *iscobol.gui.windows\_darkening=15* the window shown in Figure 12, *Window before darkening effect*, has its normal color during the accept, but when the message box appears, the color is automatically changed and shown in in Figure 13, *Window after darkening effect*.

Figure 12. Window before darkening effect

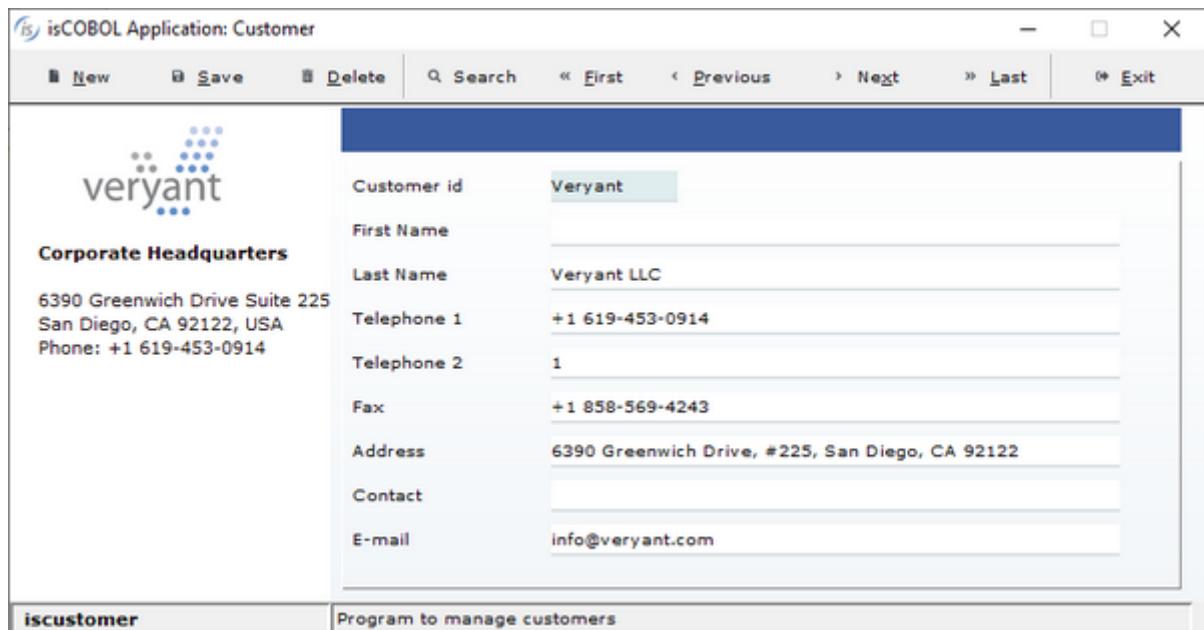
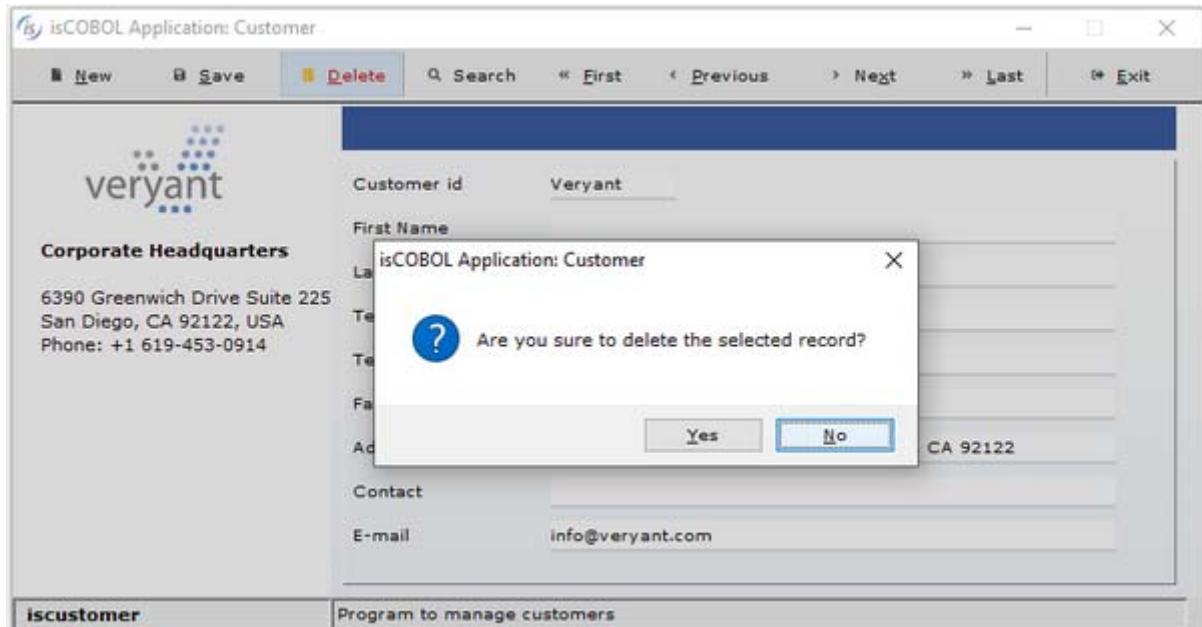


Figure 13. Window after darkening effect



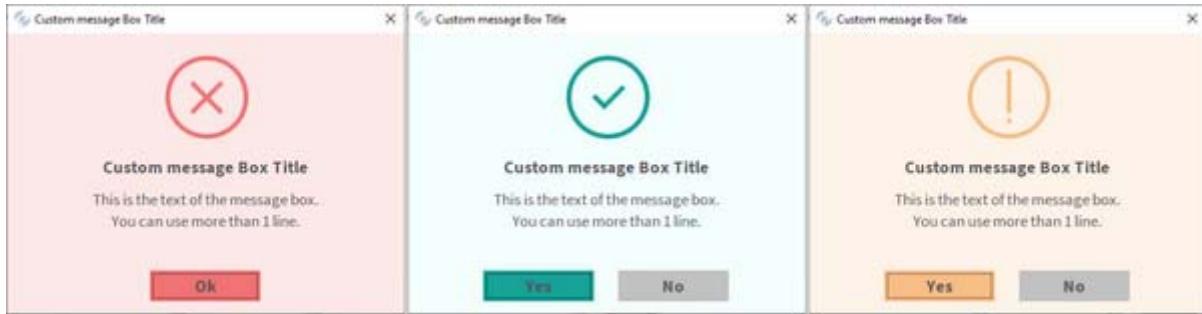
- *iscobol.gui.layout\_manager.max\_font\_zoom=n* to specify the upper limit for the font increase performed by the LM-ZOOM layout manager
- *iscobol.gui.layout\_manager.min\_font\_zoom=n* to specify the lower limit for the font reduction performed by the LM-ZOOM layout manager
- *iscobol.gui.light\_gray\_is\_transparent=false* to avoid the transparent effect to be applied automatically on color 0xc0c0c0

Additionally, in the existing configuration *iscobol.gui.messagebox.custom\_prog*, the value of the program name can be followed by comma and a letter to specify the program location. When 'C' is used, the program is searched for in the client machine, otherwise it's searched for in the server machine. For example, configuring:

```
iscobol.gui.messagebox.custom_prog=CUSTOMMSG,S
```

the Cobol program named CUSTOMMSG is called when programs display a MESSAGE BOX, and this program is searched for on the Server. Figure 14, *An example of a custom message box*, shows some message box fully customized.

Figure 14. An example of a custom message box



## isCOBOL Compiler

The isCOBOL 2022R1 compiler includes improved compatibility with other COBOL dialects to simplify migrations, and isCOBOL ESQL has been improved as well to ease migration from the IBM DB2 PRE (ESQL precompiler).

### Improved Compatibility with other COBOLs

Enhancements have been made in this release to improve our compatibility with MicroFocus COBOL, such as:

- WITH DEBUGGING MODE in SOURCE-COMPUTER is now supported to handle code lines marked with D. For example, this code snippet:

```

CONFIGURATION SECTION.
SOURCE-COMPUTER. WITH DEBUGGING MODE.
working-storage section.
...
procedure division.
MAIN.
D   DISPLAY "LOG: calling ELCUST"
      CALL "ELCUST" USING PARAMS
D   DISPLAY "LOG: returned from ELCUST"

```

turns on debugging mode, causing the lines containing the DISPLAY statements to be included in the compiled program. If the WITH DEBUGGING MODE statement is commented out, like this:

```
SOURCE-COMPUTER. . *->WITH DEBUGGING MODE.
```

all lines marked with D in the Indicator area, column 7 for ANSI sources, are treated as comments and the DISPLAY statements will not be included in the compiled program.

- the XML syntax extension with ORGANIZATION IS XML in SELECT and XD for record definition is supported to clean compile.

A code such as:

```
FILE-CONTROL.  
select xml-customer assign "out-customer.xml"  
      organization is xml  
      document-type is "customer"  
      file status is xml-status.  
file section.  
xd xml-customer.  
01 rec-customer identified by "customer".  
  05 xmlns-name pic x(80) identified by "elementName" is attribute.  
  05 xmlns-age  pic 9(18) identified by "elementAge".  
...  
  open output xml-customer  
  write rec-customer  
  close xml-customer
```

is now compiled without needing compiler options, enabling reading and writing XML files.

- the direct syntax is now supported on INVOKE statement, using the code snippet:

```
invoke JSystem::"getProperty" ("user.home") returning w-home
```

or:

```
invoke JSystem:>"getProperty" ("user.home") returning w-home
```

are equivalent of:

```
invoke JSystem "getProperty" using "user.home" returning w-home
```

Enhancements have been made to improve the compatibility with RM/COBOL, such as:

- a new compiler option, -crko, has been implemented to declare keys of indexed files in offset order including segments. This option is useful to have the same file definitions of indexed files, both when isCOBOL is used to access existing RM indexed files through RMC (RM file Connector) or when using the ISMIGRATE utility to migrate the existing files to the desired Veryant file system such as C-Tree or JISAM.
- enhanced the existing -cr option to better manage the character DISPLAY with POS declared without LINE.

An enhancement in the compiler has been implemented to improve the compatibility with ACUCOBOL-GT:

- the COPY RESOURCE directive is now fully supported. It allows you to embed any file in the class. This is typically useful to load an image using the W\$BITMAP routine that is not present on the disk and needs to be embedded in the program class. Code like the following:

```
copy resource "..\resources\logo.png".  
call "w$bitmap" using wbitmap-load, "logo.png"  
      giving h-bmp.
```

will load the image "logo.png" and embed it in the class at compile time.

## Improved Compatibility with DB2 ESQL Precompiler

The Rowset positioning syntax is now supported by ESQL cursors, to simplify the migration from DB2 Precompiler to the isCOBOL compiler, that supports ESQL statements accessing directly with the JDBC driver. This feature enables programs to read one or more records from a cursor starting either from the beginning of the cursor or from a given position in the cursor. As an example, given the following table:

CUST_ID	CUST_NAME
1	Daniel
2	Robert
3	Bill
4	John
5	Richard
6	Eleonore
7	Bob
8	Cindy
9	Rachel

Create a cursor with the following syntax:

```
exec sql
declare cur cursor
  with rowset positioning
    for select * from customers
end-exec
```

After opening the cursor, you can read record 4 (John) and 5 (Richard) with the following syntax:

```
exec sql
  fetch rowset
    starting at absolute 3
  cur
    for 2 rows
      into :wrk-id, :wrk-name
end-exec
```

You can then read records 7 (Bob), 8 (Cindy) and 9 (Rachel) with the following syntax:

```
exec sql
  fetch rowset
    starting at relative 1
  cur
    for 3 rows
      into :wrk-id, :wrk-name
end-exec
```

The ABSOLUTE clause counts records from the beginning of the cursor while RELATIVE counts from the current position in the cursor.

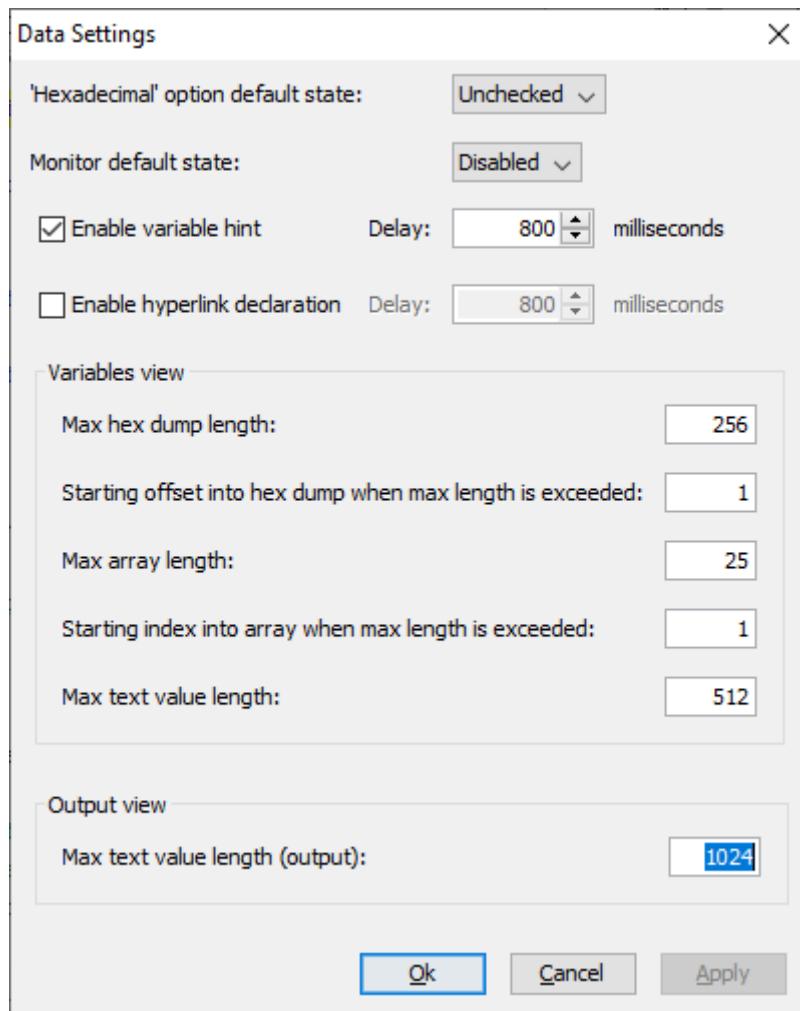
The wrk-id and wrk-name data items should be OCCURS items as they will host multiple results.

## isCOBOL Debugger

The isCOBOL Debugger has been enhanced in the 2022R1 release to better manage the Output view. These are the features added:

- A new Data Setting is available to set the max text length in the Output view. This affects the double click on the data item or the command-line DISPLAY command, since both show the content of the variable in the Output view. When a variable has a long text (such as a PIC X ANY LENGTH or a PIC X(5000), it can be useful to define a limit to reduce the text shown. If the text is longer, then the characters “...” are shown after the limit. As shown in Figure 15, *Max text value length*.

Figure 15. Max text value length



- Data items can now be selected using the keyboard in the selected line pressing Tab or Shift+Tab. This makes it easy to use using the Debugger with the keyboard. Tab moves the selection to the next data item, while Shift+Tab moves the selection to the previous one.

For example, when selecting a source line that contains:

```
move w-customer-code    to fd-customer-code
```

it is now possible to press Shift+Tab (or Tab 2 times) to select the second data item. Pressing Ctrl+D or Ctrl+Q will cause the “Display variable” window or the “Quick Watch” window to appear with variable name fd-customer-code already entered in the field used to specify the variable name.

- Better focus management of the Debugger command line area. When the focus is moved from the Debugger command line area to another view, such as Perform stack, Breakpoints, Monitors, Current Variables, there is no need to move the focus back again to the command line area to insert a new command. In every Debugger view, when pressing an alphabetic character for commands, like D to insert the Display command or M to insert the Monitor command, the focus is automatically given to the command line area and the pressed character is input to complete the command.

## isCOBOL EIS

isCOBOL EIS, Veryant’s solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. In isCOBOL 2022R1, the HttpClient class can set and get timeouts, the \$ELK directives used to generate the ServiceBridge programs have been enhanced and new configuration settings are available.

### HttpClient

HttpClient is a class that enables COBOL programs to interact with Web Services. This class has been updated to manage the Timeout during connections and reads.

The new method signatures are shown below:

- public void setConnectTimeout(java.lang.Double)
- public void setReadTimeout(java.lang.Double)
- public java.lang.Double getConnectTimeout()
- public java.lang.Double getReadTimeout()

If the connect or read operation is not completed within the timeout value, an exception will be raised. This can be used to avoid having a program hangs if a URL is not responding, and without needing to add additional code to avoid this situation.

### \$ELK directives

When generating service bridge programs, \$ELK directives help to better describe the operations and parameters that need to be passed when consuming web services. In this release, a new directive named \$ELK NULLABLE is supported to set a field value to null in the Json stream. Also, the IDE Service Bridge editor has been updated to give the ability to generate this new directive in the linkage section of Cobol source.

An example is shown in Figure 16, *Nullable in Service Editor*.

The Nullable check-box has been checked on the output field resp2, so in the Cobol source the linkage section will be updated as:

```
$elk output
$elk nullable
05 resp2 pic 9(3) .
```

and in the generated program restwebservice-req1.cbl, the Json definition of the response will be:

```
01 response-varout identified by "Response".
02 request1-out identified by "request1_out".
06 RET-STATUS-outvar identified by "ret_status".
07 ret_status-out pic x any length.
06 RESP1-outvar identified by "resp1" is nullable.
07 resp1-out pic x any length.
06 RESP2-outvar identified by "resp2" is nullable.
07 resp2-out pic 9(3) .
```

COBOL programs can then set null values in the field's initialization, instead of using spaces for pic x data items, or 0 in pic 9 data items.

The Json stream returned to the client consumer of this REST web service will be:

```
{  
  "Response":{  
    "request1_out":{  
      "ret_status":"OK",  
      "resp1":null,  
      "resp2":null  
    }  
  }  
}
```

In addition, the existing base64Binary and hexBinary values supported in the \$elk type directive (previously supported only by SOAP web services) can now also be set when using REST web services. For example, the code snippet:

```
01 request1 identified by "request1".  
$elk input  
$elk type=base64Binary  
  05 field1 pic x any length.  
$elk output  
$elk type=hexBinary  
  05 field2 pic x any length.
```

defines a Json stream structure where the input to field1 contains a base64Binary parameter type, and the output to filed2 is returned as hexBinary type.

## Configuration settings

New configuration settings have been implemented to customize the initialization of items in the stream structure:

- *iscobol.xmlstream.initialize\_on\_read=true* to initialize the data item associated to the xml stream object
- *iscobol.jsonstream.initialize\_on\_read=true* to initialize the data item associated to the json stream object

## Additional improvements

isCOBOL 2022R1 improves several utilities:

- in GIFE, the Graphical Indexed and relative File Editor utility, a new configuration has been added:
  - o *iscobol.gife.custom\_font=FontName-size* to customize the font used in the GIFE user interface. By default, it loads the default Font of the current LookAndFeel used, which varies based on the options –system, --nimbus, --metal, etc.
- in JDBC2FD, the utility that enables you to connect to RMDBS database and generate the COBOL copy files (.sl for the Select, .fd for the File Definition, .wrk for FD/Working structure), a new configuration has been added:
  - o *iscobol.jdbc2fd.current\_schema\_only=true* to load only the items of the current schema by JDBC2FD. This simplifies the filter on table names loaded after the connection.

# isCOBOL Evolve

---

## isCOBOL 2021 Release 2 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2021 R2.

The new 2021 R2 release has been updated to allow greater integration of isCOBOL applications running in WebClient and HTML / JavaScript components.

New compatibility enhancements provide greater flexibility when porting from other dialects.

Details on these enhancements and updates are included below.

### WebClient integration with a web page

WebClient is Veryant's solution to host desktop application in a web environment. Desktop applications will be rendered as HTML pages and will run unchanged in a browser. Using WebClient a desktop application can be embedded inside a custom-built web page to provide additional functionality.

With the 2021R2 release, the range of capabilities have been greatly expanded. isCOBOL applications can now interact with the underlying web page, and the page can invoke isCOBOL code. Additionally, a new IWC-PANEL component has been implemented to host web components inside COBOL screen sections.

Communications is based on messages that COBOL and the web page can exchange to perform tasks.

A message is a data structure composed of:

- *action*: a string containing the purpose of the message, and is required
- *data*: a string with the parameters for the action, an optional parameter
- *binaryData*: a byte array with parameters for the action, an optional parameter

COBOL programs have access to the following new routines:

- IWC\$INIT to activate the communication between COBOL and the web page
- IWC\$GET to read the data sent by the Javascript code in the web page
- IWC\$SEND to send data to the web page
- IWC\$STOP to stop the communication between COBOL and the web page

To embed a COBOL program in a web page when running in WebClient, thus enabling communication, a container web page needs to be provided, and the "Compositing Window Manager" setting in the WebClient configuration for the COBOL application needs to be enabled.

The web page needs to contain, as a minimum, the following code that defines a div element that will contain the COBOL application:

```
<div class="webclientAppContainer webswing-element"
      data-webswing-instance="webclientInstance">
    <div id="loading" class="ws-modal-container">
      <div class="ws-login">
        <div class="ws-login-content">
          <div class="ws-spinner">
            <div class="ws-spinner-dot-1"></div>
            <div class="ws-spinner-dot-2"></div>
          </div>
        </div>
        </div>
      </div>
    </div>
</div>
```

The data-webswing-instance tag specifies the JavaScript variable that will hold a reference to the COBOL instance running in WebClient. The object has an options property that can be used to handle the interaction:

```
var webclientInstance = {
  options: {
    autoStart: true,
    args: '',
    recording: getParam('recording'),
    debugPort: getParam('debugPort'),
    connectionUrl: '<URL of the webapp as defined in WebClient>',
    compositingWindowsListener: {
      windowOpening: function(win) {},
      windowOpened: function(win) {},
      windowClosing: function(win) {},
      windowClosed: function(win) {},
      windowModalBlockedChanged: function(win) {}
    },
    customization: function(injector) {
      injector.services.base.handleActionEvent =
        function(actionName, data, binaryData) {
          if (actionName == "action") {
            /* code to handle the action */
          }
        }
    }
  }
}
```

When WebClient loads the web page, it will enrich the webclientInstance object with additional properties and methods useful for interacting with the COBOL program.

The handleActionEvent callback is an event handler that will be called when the COBOL program executes the IWC\$SEND routine, and any needed code to carry out the requested action can be added there.

To send an action from Javascript to the COBOL program the following code can be used:

```
webclientInstance.performAction (
{actionName: 'EXECUTE_PGM', data: 'INVOICE_PRINT', binaryData: null}
)
```

The action details can be retrieved in COBOL by calling the IWC\$GET routine.

The following code shows the COBOL side of the communication:

```
78  iwc-crt-status      value 1001.
77  data-to-send        pic x any length.
01  iwc-struct.
    03  iwc-action       pic x any length.
    03  iwc-data         pic x any length.
    03  iwc-bytes        pic x any length.

ACTIVATE.
  call "IWC$INIT" using 78-iwc-crt-status
                giving return-code.

SEND-TO-HTML.
  initialize iwc-struct.
  move "ComSample" to iwc-action
  move data-to-send to iwc-data
  call "IWC$SEND"   using iwc-struct
                giving return-code.

READ-DATA-FROM-HTML.
  initialize iwc-action
  call "IWC$GET"   using iwc-struct
                giving return-code
  if iwc-action = "EXECUTE_PGM"
    call IWC-DATA
  end-if.

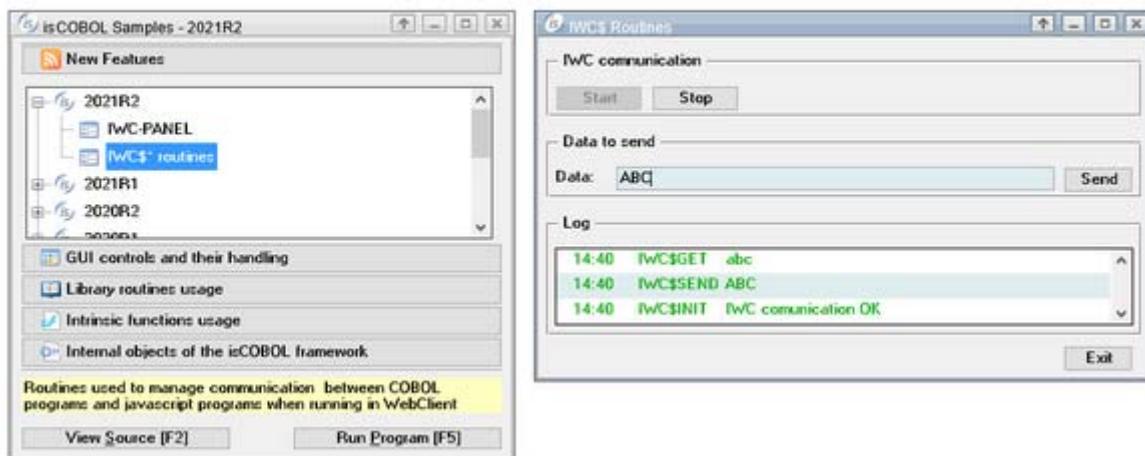
DEACTIVATE.
  call "IWC$STOP" giving return-code.
```

Every time the web page sends a message to the application, the COBOL program can read it calling the IWC\$GET routine. If a program is executing an ACCEPT statement, it will be terminated with the key-status specified in the IWC\$INIT routine.

To send a message to the web page, the IWC\$SEND can be called passing the same message structure iwc-struct described above. Communication can be stopped by calling the IWC\$STOP routine.

The result of program running in WebClient is shown in Figure 1, IWC\$ routines.

Figure 1. IWC\$ routines



The window on the right shows the result of calling a JavaScript function that transforms the string it receives to lower case as the action data parameter and sends it back to the COBOL program using the `performAction` method. The COBOL program can read the result using the `IWC$GET` routine.

### Using Web Components in COBOL screen sections.

Starting from the 2021R2 release, custom HTML / JavaScript web components can be embedded in COBOL screen sections when running in WebClient, allowing the creation of hybrid apps that were not possible before. The feature is very powerful yet easy to use. On the COBOL side of the program, only COBOL knowledge is required, while on the web page and component creation and handling, HTML / Javascript and CSS knowledge is required, as development will be done using a web toolchain.

To host a Javascript component, a new `IWC-PANEL` control has been implemented for the SCREEN SECTION. The component is only visible when the application is run in a WebClient environment, and will be ignored when running as a desktop application.

`IWC-PANEL` acts as a placeholder in the HTML page, and the actual content will need to be injected from the HTML / Javascript page, just like it's done in traditional web applications.

An example of the panel is provided in the code below:

```
03 f-map iwc-panel
      js-name      "f-map"
      line 5       column 2
      size 68 cells lines 15 cells
      value        fmap-struct
      event procedure FMAP-PROC.
```

The `VALUE` property of the control holds the message structure used to send actions to the panel in the web page. The message is sent by performing a `MODIFY` statement on the `value` property. The event procedure will be called when the web page executes a `performAction` on the panel, and an `INQUIRE` on the `value` property of the `IWC-PANEL` will return the message that has been sent.

The JS-NAME property holds an identifier that will be sent to the web page upon creation, so that the corresponding web component can be created. For every IWC-PANEL in a form, a callback in the web page is called, with the details necessary to perform component initialization. The webclientInstance.options.compositingWindowsListener object defines callbacks for various events, ranging from windows opening, closing and IWC-PANEL creation.

An IWC-PANEL creation will trigger the windowOpened callback, and a reference to the IWC-PANEL is passed as function argument. The callback can check the .name property to determine which control has been created and react accordingly.

A sample code snippet is:

```
...
compositingWindowsListener: {

windowOpened: function (win) {
    if (win.name == 'map') {
        createMap(win);
    }
}
...
}
```

The webclientInstance object has a getWindows() method that will return all windows and IWC-PANEL that the COBOL application has created, along with the DOM (Document Object Model, the in-memory representation of the HTML page created by the browser) element of each.

A sample project is provided that shows how to integrate a Google map component in a COBOL application, and how to interact with it.

### How to integrate a Google map component

This code snippet below is taken from the Google maps integration sample, and shows the code in WORKING-STORAGE that defines the structure for the messages to be sent to and received from the web, the definition of the new IWC-PANEL control in the SCREEN SECTION and the PROCEDURE DIVISION showing sample code to invoke actions in the web page to perform specific tasks and handle incoming messages.

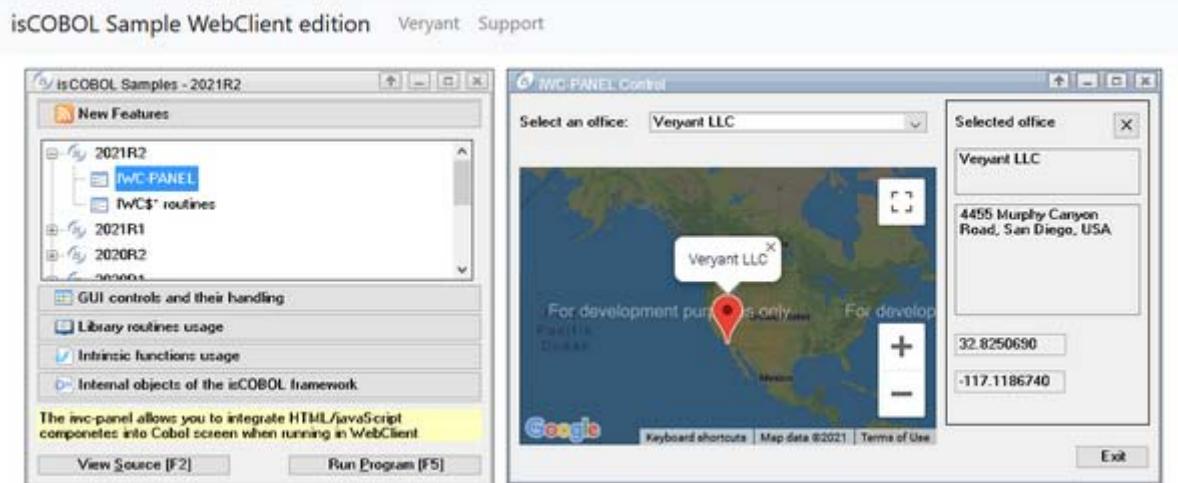
The snippet shows the interaction with a Google map element created in the page, and how to send JSON (the native Javascript data format) data as argument of the action. When the user selects an office from the COBOL combo-box, the MODIFY statement is executed on the IWC-PANEL, causing a "selectOffice" action with a JSON representation of the selected office to be sent to the web page, and the Javascript code on the page will center the map on the requested office location.

When the user clicks on a pin in the Google map, the Javascript program calls the panel's performAction method, causing the IWC-PANEL event procedure to be called. Performing an INQUIRE on the value property of the panel will return the data structure sent by the JavaScript code.

```
WORKING-STORAGE SECTION.  
01 fmap-struct.  
  03 fmap-ACTION  PIC X any length.  
  03 fmap-DATA    PIC X any length.  
  03 fmap-BYTES   PIC X any length.  
  
SCREEN SECTION.  
01 Mask.  
  03 f-map iwc-panel  
      js-name          "f-map"  
      line 5           column 2  
      size 68 cells    lines 15 cells  
      value            fmap-struct  
      event procedure  FMAP-PROC.  
  
...  
SHOW-ON-MAP.  
  move "selectOffice"      to fmap-action  
  move offices(office-index) to selected-office  
  set objJsonStream to jsonStream:>new(selected-office, 1);;  
  set strbuffer      to string-buffer:>new  
  objJsonStream:>writeToStringBuffer(strbuffer)  
  move strbuffer:>toString to fmap-data  
  modify f-map value fmap-struct.  
  
FMAP-PROC.  
  if event-type = ntf-iwc-event  
    inquire f-map value in fmap-struct  
    evaluate fmap-action  
    when "pinClicked"  
      move fmap-data to sel-description  
    ...  
    when "pinClosed"  
    ...  
    end-evaluate  
  end-if.
```

The result of program running in WebClient is shown in Figure 2, IWC-PANEL control.

Figure 2. IWC-PANEL control



The new features let developers easily create a web portal with an HTML menu system that starts COBOL programs hosted in containers inside the main page. Several programs can be started at once and can be managed using the features described above. A sample menu application is included in the installation, that uses Iframes to hold each instance of COBOL programs, and can be easily inspected and customized to provide a full-blown web solution for legacy applications. The sample is an evolution of our trusty old isapplication sample, fully evolved to be a modern web application.

Taking some care when styling the COBOL application will make the end result look like a complete native web application, with the advantages of having a familiar COBOL development environment to provide modernization of code.

Figure 3, Web portal and WebClient, shows the result of running the new provided sample that uses the techniques described above to provide an integrated environment with HTML code and COBOL programs running together seamlessly.

Figure 3. Web portal and WebClient

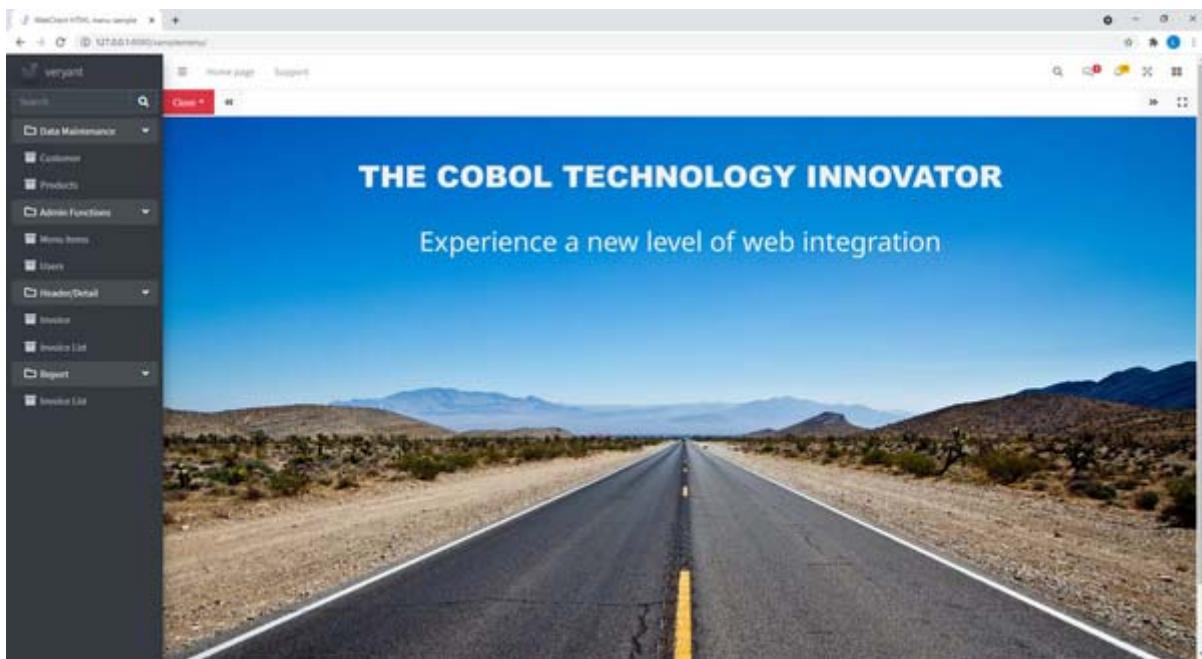


Figure 4, HTML / JS and COBOL SCREEN, shows an HTML / JS menu that run COBOL programs with SCREEN SECTION styled to look like native web applications.

Figure 4. HTML/JS and COBOL SCREEN

Product ID	Name	Description	Price
MG5201	CorelDraw 12 (S)	COREL CorelDraw Graphic Suite 12 (S)	\$9.99
MG5202	CorelDraw 12 (F)	COREL CorelDraw Graphic Suite 12 (Full)	\$9.99
MG5203	CorelDraw 11 (F)	COREL CorelDraw Graphic Suite 11 (Full)	\$9.99
CL101	Cubasis VST 3.0 Pinnacle Cubasis VST 3.0	\$9.99	
MG5204	Wacom Draw Master MG5-graphic tablet Draw Master 20x15c	\$9.99	
MG5205	Wacom Cintiq 14 MG5-graphic tablet Cintiq 14x10cm 1024x1024	\$9.99	
SONY101	SONY T2XP/S	SONY T2XP/S	\$9.99
W101	INTUOS3	INTUOS3	\$9.99
W102	INTUOS3 S/M US	INTUOS3 S/M US	\$9.99
W103	INTUOS3 M/L US	INTUOS3 M/L US	\$9.99
W104	INTUOS3 L/XL US	INTUOS3 L/XL US	\$9.99
W105	VOLUT 2 M/L US	VOLUT 2 M/L US	\$9.99
W106	INTUOS3 S/M US WACOM graphic tablet INTUOS3 S/M US	INTUOS3 S/M US WACOM graphic tablet INTUOS3 S/M US	\$9.99
W107	INTUOS3 M/L US WACOM graphic tablet INTUOS3 M/L US	INTUOS3 M/L US WACOM graphic tablet INTUOS3 M/L US	\$9.99

## isCOBOL Compiler

Starting from the isCOBOL 2021R2 release, the compiler option -sp, used to set the location of copy files, is not needed to include standard isCOBOL copy files. When a COBOL source refers to an isCOBOL .def file, the compiler automatically looks for it in the isCOBOL\_SDK sample/isdef folder.

The above folder is also searched for even if the -sp option is supplied, but a .def file is not found in the specified location.

The ANY LENGTH data item can now be used in LINKAGE SECTION items in the called program, even if the caller program passes a fixed length data item, allowing for more flexibility. An item declared as ANY LENGTH in the LINKAGE SECTION will accept both fixed length or variable length parameters, and if the caller passes a fixed length data item, the called program will treat it as a fixed length size.

The ON EXCEPTION clause is now supported on SET ENVIRONMENT statements to catch the failed SET on sticky configurations. This is useful, for example, when the Application Server loads its configuration that should not be overwritten by a ThinClient application that loads a remote configuration file.

For example, if the server starts with the configuration:

```
iscobol.as.authentication=1
```

and a program running in ThinClient executes:

```
set environment "as.authentication" to "0"
on exception
    display "setenv failed"
end-set
```

the ON EXCEPTION clause will be executed as the SET statement of this sticky configuration fails.

## Compatibility with other COBOLs

To enhance the compatibility with MicroFocus and IBM COBOL, a new compiler configuration iscobel.compiler.command\_line\_linkage=true can be used to preload a linkage data item in the main program. This allows passing a command line parameter directly to a linkage data item in the executed program, and the parameters received in chaining are automatically set in the LINKAGE SECTION.

Also, a compiler directive \$SET is now available to specify this feature on a specific COBOL source program, for example:

```
$SET "COMMAND_LINE_LINKAGE" "1".
PROGRAM-ID. MY_PROG .
```

A possible use of this feature, besides compatibility with other dialect, is when running a batch program that is usually called from another COBOL program, which passes needed arguments in the LINKAGE SECTION. If such program needs to be run stand-alone from the command line, using the above option will allow to specify the arguments in the command line, and they will be copied from the chaining parameters to the LINKAGE SECTION, allowing it to run unmodified.

Binary and Octal values can now be also specified using the MicroFocus syntax b"binaryValue" and o"octalValue" when compiling with -cm compiler option. Without this option the isCOBOL compiler only supports the equivalent syntax B#binaryValue and O#octalValue.

This is a snippet of code now supported:

```
78 CONST-BINARY value B"101".
78 CONST-OCTAL  value O"12345670".

MOVE B"101"      TO MYVAR1.
MOVE O"12345670" TO MYVAR2.
```

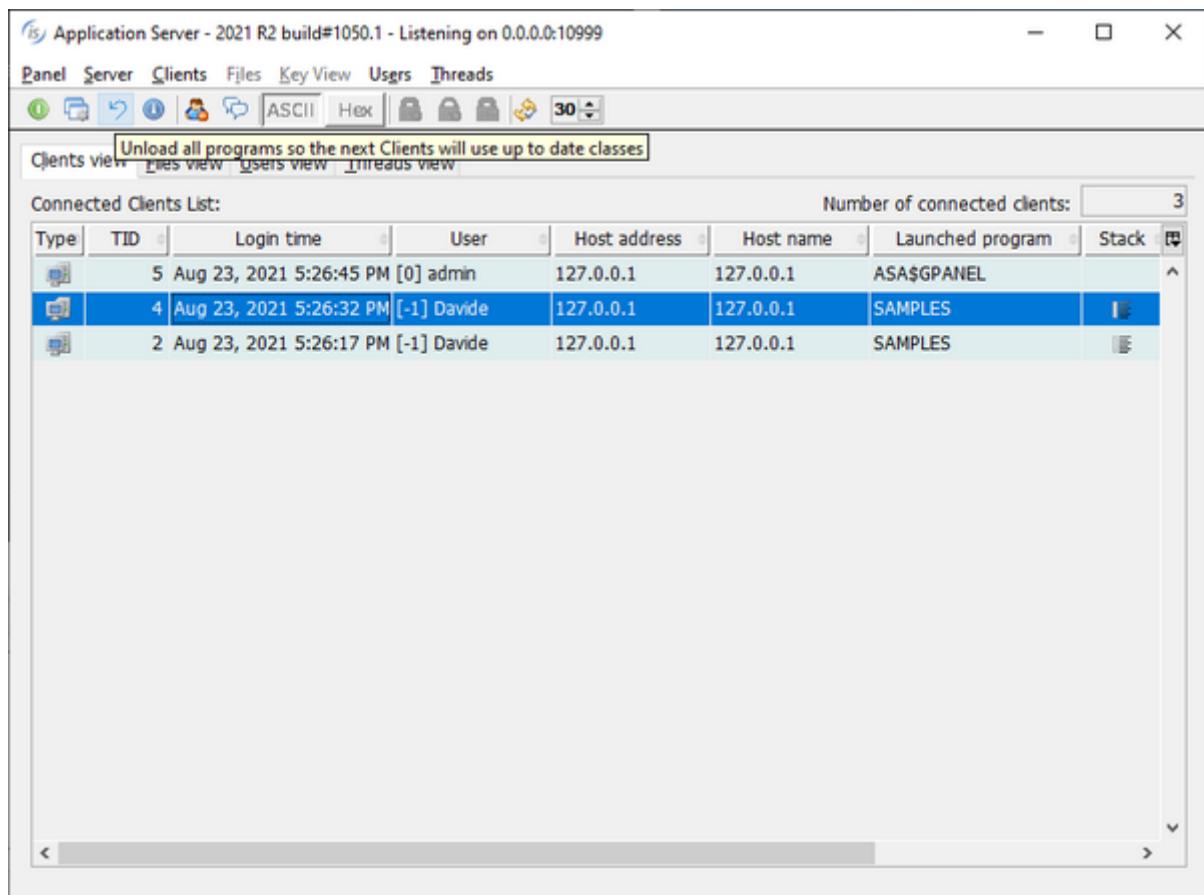
## isCOBOL Runtime

In the 2021R2 release several new configuration settings have been added:

*iscobol.code\_prefix.reload=2* can be used to reload the whole set of classes specified in the *code\_prefix* configuration variable after calling the C\$UNLOAD library routine. This option can be used to automatically reload classes in a ThinClient installation after an upgrade, to ensure that all classes are updated.

The isCOBOL Server Panel has been upgraded to allow reloading of classes for applications running with the new configuration variable. The new button to force a reload is located in the tool-bar, as shown in Figure 5, isCOBOL Panel with new reload logic.

Figure 5. Web isCOBOL Panel with new reload button



New configurations have also been implemented to allow print jobs to be executed asynchronously. When a CLOSE print-file statement is executed, a thread that manages the entire print job is created, allowing execution to continue in the application so that users can continue to work without needing to wait for the print job to complete. The new configurations are:

*iscobol.print.spooler\_async=true|false* (default: true) to set the print job to be run asynchronously

*iscobol.print.pdf\_async=true|false* (default: false) to have the PDF print job to be executed asynchronously.

Performance of print jobs have been improved, both as a result of the async mode implementation and as a better Application Server TCP packet handling when running in ThinClient.

Figure 6, Print jobs comparison, shows the gains of printing using the new 2021R2 release compared to the previous 2021R1.

Figure 6. Print jobs comparison

	2021R1	2021R2
<b>Standalone tests</b>		
character print job with 200 pages on -P SPOOLER	18,27	6,35
graphical print job with 20 pages on -P SPOOLER	11,37	4,38
<b>ThinClient tests</b>		
character print job with 200 pages on -P SPOOLER	16,93	1,75
graphical print job with 20 pages on -P SPOOLER	69,80	14,75
character print job with 200 pages on -P PDF	17,53	13,70
graphical print job with 20 pages on -P PDF	64,86	22,05
character print job with 200 pages on -P PREVIEW	6,79	2,04
graphical print job with 20 pages on -P PREVIEW	59,10	14,25

All times are in seconds.

Hardware details of client machine:

Windows 10 Pro i7-8550U CPU @ 1.80GHz 16GB

Hardware details of server machine:

macOS Big Sur Apple M1 16GB.

## isCOBOL EIS

HTTPClient is a class that provides many useful features to communicate with existing HTTP services like Web Service (REST/SOAP) HTTP servers, and HTTPHandler is a class that provides a communication bridge between COBOL programs and HTML5/JavaScript pages using the HTTP protocol. Both classes have been upgraded with new features.

### HTTPClient

The HTTPClient class now allows you to specify the Encoding charset in the following methods:

- `displayEx ( stream, hasDummyRoot, charset )`
- `displayJSON ( json, hasDummyRoot, charset )`

The dummyRoot parameter is now supported in the getResponseJSON method:

- `getResponseJSON( json, encoding, hasDummyRoot )`

`hasDummyRoot` is an alphanumeric data item or literal hosting a Boolean value. If the Boolean value is TRUE, then the top-level item of Record-Definition is discarded and will not appear in the JSON stream

## HTTPHandler

The dummyRoot parameter is now supported in the acceptEx and acceptFromJSON methods:

- `acceptEx( params, hasDummyRoot )`
- `acceptFromJSON( params, hasDummyRoot )`

## isCOBOL Evolve

---

# isCOBOL 2021 Release 1 Overview

## Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2021 R1.

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

Working with remote COBOL applications is now fully integrated in the isCOBOL IDE project environment.

The 2021R1 release has new GUI features that help developers modernize applications.

The new version of the IsCOBOL compiler offers better compatibility with IBM COBOL compilers.

WebClient, Veryant's solution to run desktop COBOL applications in the browser, has been vastly updated in its core technologies, adding new features and capabilities that simplify deployment and administration in clustered cloud environments.

The new C-Tree RTG v3 is now included in the 2021 release.

Details on these enhancements and updates are included below.

## isCOBOL IDE enhancements

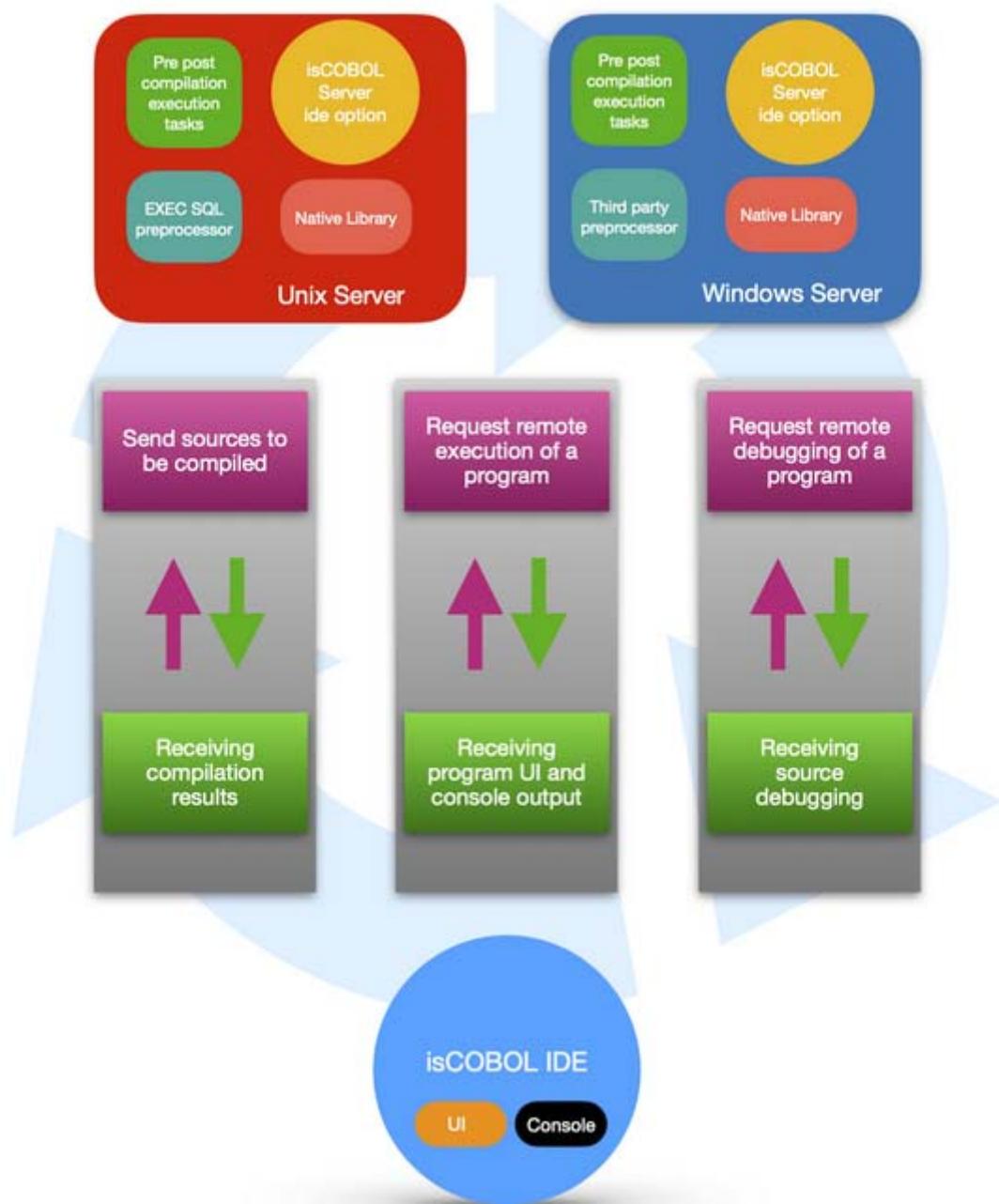
The isCOBOL IDE, now based on Eclipse 2020-06 (4.16), offers a new feature to easily work with remote projects. Remote projects are defined by having local source code, but needing remote compilation and execution. Typical scenarios for remote projects are batch programs that need to be executed on a specific server, for instance if they make use of native calls or third-party pre-compilers. The IDE now allows you to manage such projects, allowing development on a desktop Operating System (Windows, MacOS or Linux), but actions such as compile, run and debug are executed remotely on a target server.

The layout of the Compile / Runtime options has been upgraded for remote projects, and also support local projects.

### Remote projects in the IDE

The isCOBOL IDE can now compile and run programs on a remote server instead of the development PC. This feature is particularly useful when you have to interact with specific software that might not be installed on your local PC but is available on a dedicated server machine.

Figure 1. Remote projects in the IDE architecture



To take advantage of this feature, the isCOBOL Server must be started with the `-ide` option on the remote server, e.g.

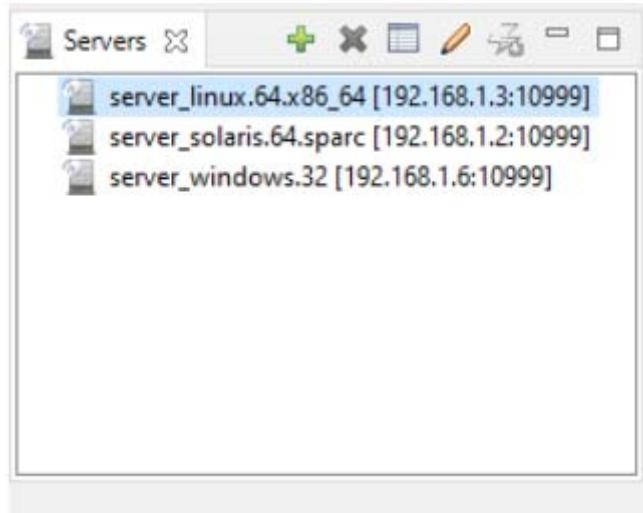
```
iscserver -ide
```

When started with the `-ide` option, the isCOBOL Server acts as a remote compiler and application server for remote IDEs.

If the isCOBOL Server is run as a Windows Service, the remote project feature can be activated by adding the `-Discobol.as.ide=true` option to the `isserver.vmoptions` configuration file.

A remote server can be registered in the isCOBOL IDE using the new Servers view, which is available in the bottom left corner of the isCOBOL Perspective. Figure 2, IDE's Servers view, shows the new view in action.

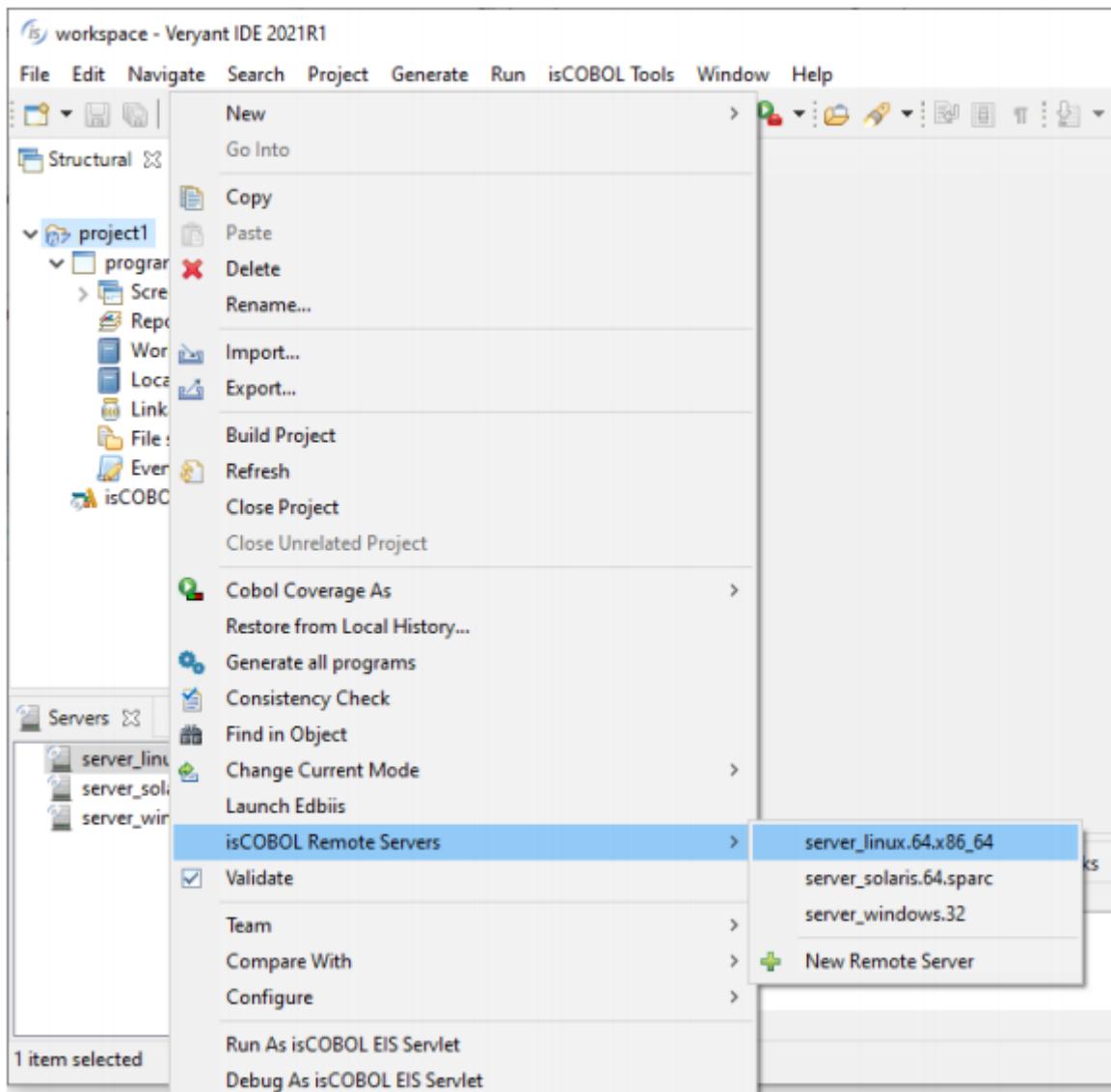
Figure 2. IDE's Servers view



To register a server, click the green + button and fill out the form with the required information: a name of your choice to identify the server in the list, and the hostname and port where the remote server is listening.

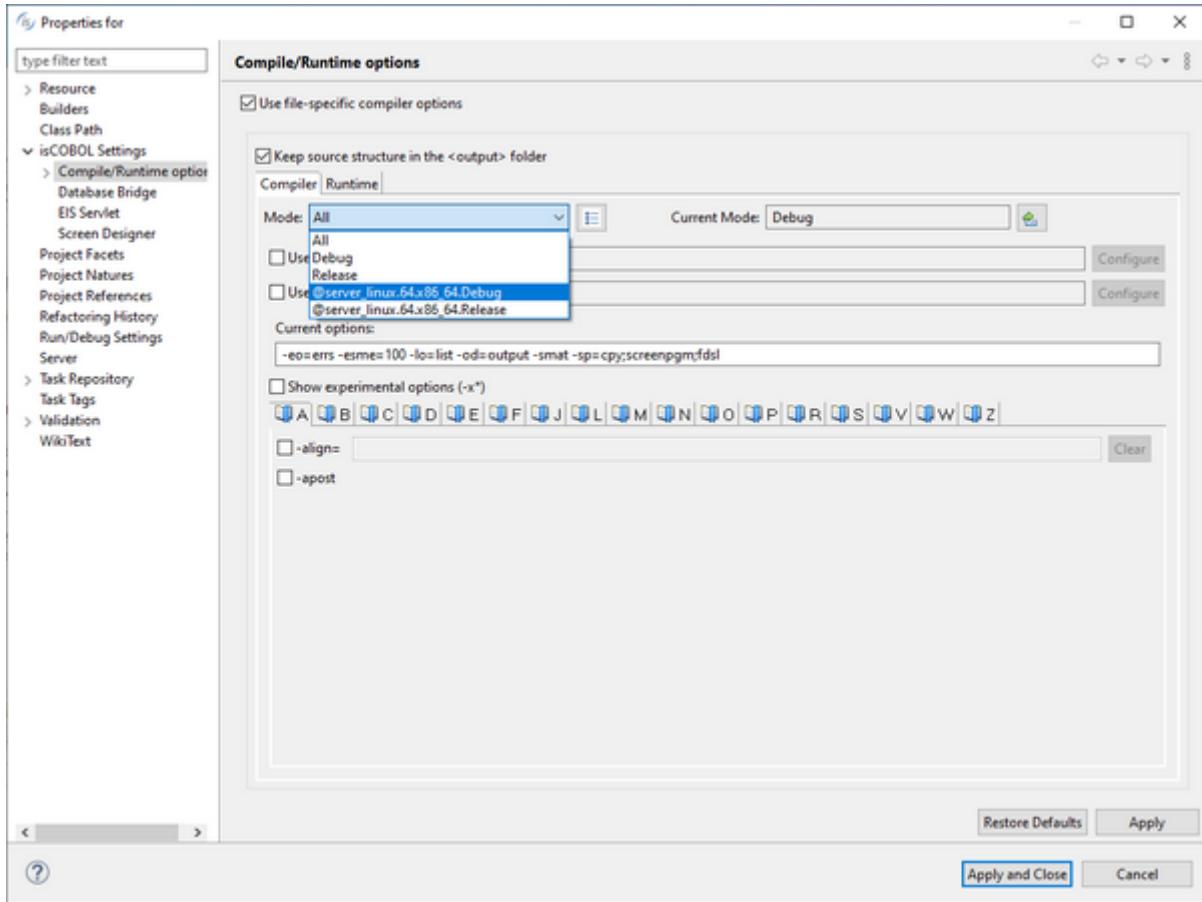
To bind a project to a remote server, right click on the project name in the isCOBOL Explorer and choose the desired server from the list in isCOBOL Remote Servers, as depicted in Figure 3, Binding a project to a remote server.

Figure 3. Binding a project to a remote server



Once the project is bound to the remote server, new compile, runtime and debug modes become available in the project properties, as depicted in Figure 4, Remote compile modes.

Figure 4. Remote compile modes



Choosing a remote mode (a mode whose name starts with @) will cause the IDE to redirect the actions to the remote server.

When you compile on a remote server, the IDE pushes the source files to the server. The source files are compiled on the server and the IDE receives the compilation results.

When you run or debug on a remote server, the IDE becomes a Thin client of the server, which results in the program being run on the server, and the GUI rendered on the local PC.

The isCOBOL Server's admin users can maintain the remote project modes through the Servers view. Administrators can edit existing projects or create new ones. isCOBOL Server's standard users can only view and use the remote project modes, but they can't edit them.

isCOBOL Server's users are maintained using the isCOBOL Server's administration panel. In Figure 5, isCOBOL Server's users, for example, two admin users (admin and pm) and two standard users (dev1 and dev2) are configured.

Figure 5. isCOBOL Server's users

The screenshot shows a software application window titled "Application Server - 2021 R1 build#1041.3 - Listening on 169.254.180.94:10999". The menu bar includes "Panel", "Server", "Clients", "Files", "Key View", "Users", and "Threads". Below the menu is a toolbar with icons for file operations like Open, Save, Print, and a search bar. The main area has tabs: "Clients view", "Files view", "Users view" (which is selected), and "Threads view". A sub-header "Users List:" is followed by a table with columns: User, Password, Admin, and ID. The table contains the following data:

User	Password	Admin	ID
admin	...	<input checked="" type="checkbox"/>	0
pm	...	<input checked="" type="checkbox"/>	0
dev1	...	<input type="checkbox"/>	1
dev2	...	<input type="checkbox"/>	1

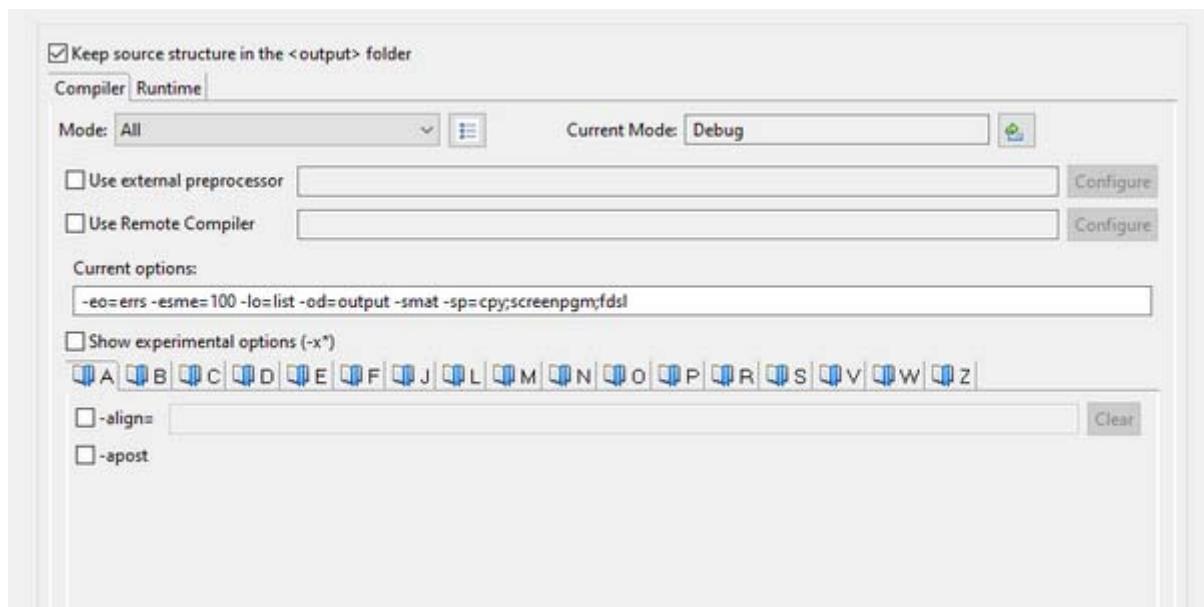
### New Layout of Compile/Runtime options

The Compile/Runtime options screen has been redesigned, separating the Compile and Runtime options.

In previous versions of the IDE, Compile and Runtime options were merged in the same Mode, and compiling a project in Release mode and running it in Debug mode required switching modes between compiling to running. Moreover, modes could not be separated between compile and run phases.

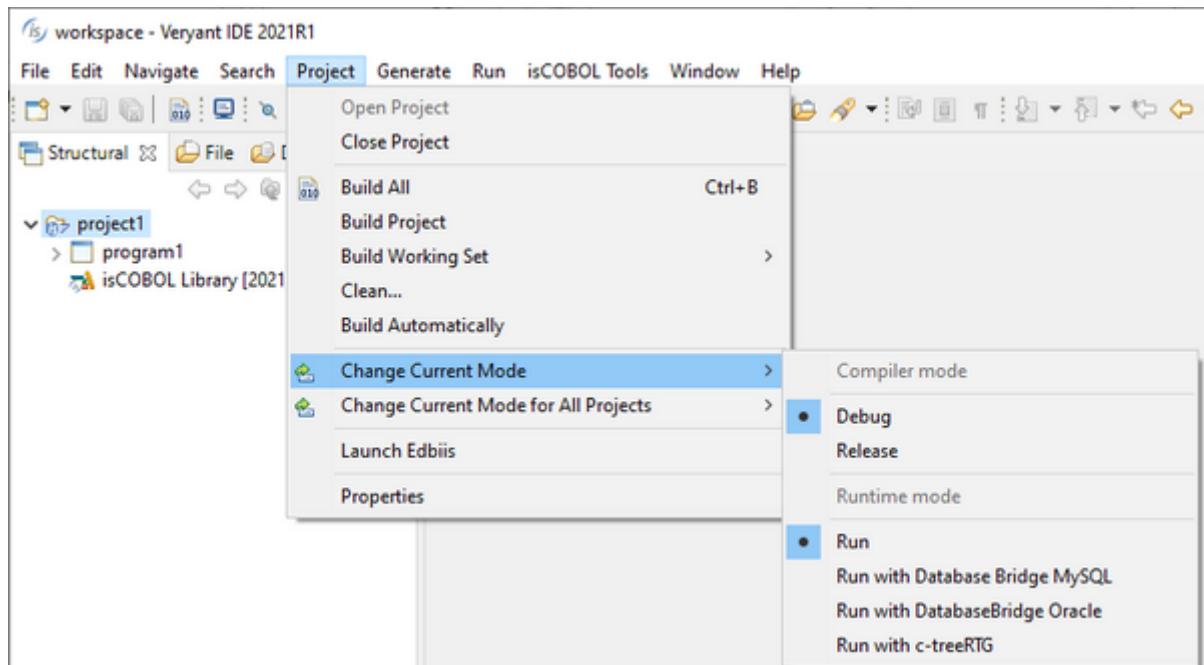
Figure 6, Compile/Runtime options, shows the separation of mode selection for the compile and runtime.

Figure 6. Compile/Runtime options



In addition, you can set the active Compile mode and the active Runtime mode in the Project -> Change Current Mode menu option; each mode will be used when compiling and running respectively. This is depicted in Figure 7, Active modes.

Figure 7. Active modes



## Better IBM COBOL compiler compatibility support

IsCOBOL Evolve 2021R1 release enhances the compiler, improving compatibility with the IBM COBOL compiler, simplifying the migration process from IBM machines to Open systems. Additionally, the ESQL syntax has been enhanced for better IBM DB2 compatibility, allowing migrations without using the db2prep pre-compiler.

### Improved IBM COBOL syntax

Compiling using the -cv compiler option now allows more IBM-specific syntax to be supported:

The USE FOR DEBUGGING declarative that allows execution of specific code while running with a configuration option set. For example, the following code snippet:

```
CONFIGURATION SECTION.  
SOURCE-COMPUTER. ... WITH DEBUGGING MODE.  
PROCEDURE DIVISION.  
DECLARATIVES.  
DEBUG-DECLARATIVES SECTION.  
  USE FOR DEBUGGING ON MYPAR1.  
DEBUG-DECLARATIVES-PARAGRAPH.  
  DISPLAY "EXECUTING: " DEBUG-NAME ", MY-VAR=" TMY-VAR.  
END DECLARATIVES.  
...  
  PERFORM MYPAR1  
...  
MYPAR1.  
  ADD 1 TO MY-VAR
```

will execute the DISPLAY statement inside the USE FOR DEBUGGING declarative every time the paragraph named MYPAR1 is executed, but only when running with the configuration option *iscobol.use\_for\_debugging=true*. Note that, following IBM rules, if the WITH DEBUGGING MODE is omitted or commented in the SOURCECOMPUTER section, the code in the USE FOR DEBUGGING declarative is never executed.

The RECURSIVE clause in PROGRAM-ID to specify that the program can be recursively called while a previous invocation is still active. Since with IsCOBOL every program can be recursively called without needing a specific declaration, the compiler treats the clause as a comment:

```
PROGRAM-ID. MYPROG IS RECURSIVE.
```

The following syntax is now supported without needing the -cv compiler option, to ease the adoption of the features in COBOL applications:

LOCAL-STORAGE SECTION in PROGRAM-ID is used to declare variables that are “local”, to prevent sharing them when recursively calling the program. WORKING-STORAGE items are global. In previous releases, the configuration option:

```
iscobol.recursion_data_global=true
```

allowed control on the working storage, which could only be either global or local.

Using the LOCAL-STORAGE SECTION allows for finer control. Below is a code snippet that shows how to declare a local variable:

```
WORKING-STORAGE SECTION.  
 77 VAR-IDX PIC 99 VALUE 0.  
LOCAL-STORAGE SECTION.  
 77 LOC-VAR-IDX PIC 99 VALUE 0.
```

New intrinsic functions named DISPLAY-OF and NATIONAL-OF to better display a National data item, declared as PIC N. The code snippet below shows the use of the new functions in MOVE and DISPLAY statements:

```
01 A-NATIONAL PIC N(1) USAGE NATIONAL.  
01 A-DISPLAY PIC X.  
...  
MOVE FUNCTION NATIONAL-OF(A-DISPLAY) TO A-NATIONAL  
...  
DISPLAY "THE CHARACTER IS: " FUNCTION DISPLAY-OF(A-NATIONAL)
```

## Improved support for ESQL on IBM DB2

A new compiler property has been introduced to inform the Compiler that the underlying database is IBM DB2.

```
iscobol.compiler.esql.db2=true
```

When this property is set to true, the Compiler generates specific code to return the result sets in the same format that would be produced when using the IBM DB2 pre-compiler. In particular, it supports the SQLDA structure and the use of date, time and timestamp as function parameters.

## SQLDA

An SQLDA (SQL Descriptor Area) is a collection of variables that are required for execution of the SQL DESCRIBE statement, and can optionally be used by the PREPARE, OPEN, FETCH, EXECUTE, and CALL statements. An SQLDA can be used in a DESCRIBE or PREPARE INTO statement, modified with the addresses of host variables, and then reused in a FETCH statement.

An SQLDA consists of four variables in a header structure, followed by an arbitrary number of occurrences of a sequence of five variables collectively named SQLVAR. In OPEN, CALL, FETCH, and EXECUTE statements, each occurrence of SQLVAR describes a variable. In PREPARE and DESCRIBE, each occurrence describes a column of a result set.

The meaning of the information in an SQLDA depends on the context in which it is used.

For DESCRIBE and PREPARE INTO, IBM DB2 sets the fields in the SQLDA to provide information to the application program. For OPEN, EXECUTE, FETCH, and CALL, the application program sets the fields in the SQLDA to provide IBM DB2 with information.

The SQLDA definition in the program Working-Storage Section is:

```
*****
* SQL DESCRIPTOR AREA
*****
01 SQLDA.
 02 SQLDAID PIC X(8) VALUE 'SQLDA'.
 02 SQLDABC PIC S9(8) COMPUTATIONAL VALUE 33016.
 02 SQLN PIC S9(4) COMPUTATIONAL VALUE 750.
 02 SQLD PIC S9(4) COMPUTATIONAL VALUE 0.
 02 SQLVAR OCCURS 1 TO 750 TIMES
    DEPENDING ON SQLN.
  03 SQLTYPE PIC S9(4) COMPUTATIONAL.
  03 SQLLEN PIC S9(4) COMPUTATIONAL.
  03 SQLDATA POINTER.
  03 SQLIND POINTER.
  03 SQLNAME.
    49 SQLNAMEL PIC S9(4) COMPUTATIONAL.
    49 SQLNAMEC PIC X(30).
```

## Date and Time functions

IBM DB2 provides a set of functions that have date, time or timestamp parameters. When the parameter value is stored in a host variable, the IBM DB2 JDBC driver would cause “invalid parameter” errors.

For example, the following query that adds 2 hours to a given timestamp fails if executed on IBM DB2 via JDBC:

```
SELECT (TIMESTAMP(:Wrk-TimeStamp) + :V2 HOURS) FROM SYSIBM.SYSDUMMY1
```

In order to make it work, a cast must be performed, e.g.

```
SELECT (TIMESTAMP((CAST(:Wrk-TimeStamp AS TIMESTAMP)) + :V2 HOURS)) FROM
SYSIBM.SYSDUMMY1
```

When `iscobol.compiler.esql.db2` is set to true, the isCOBOL Compiler takes care of applying the necessary casts in cases like the above, requiring no code changes.

The following functions are recognized by the Compiler: ADD\_DAYS, ADD\_HOURS, ADD\_MINUTES, ADD\_MONTHS, ADD\_SECONDS, ADD\_YEARS, AGE, DATE\_PART, DATE\_TRUNC, DAYNAME, DAYOFMONTH, DAYOFWEEK, DAYOFWEEK\_ISO, DAYOFYEAR, DAYS, DAYS\_BETWEEN, DAYS\_TO\_END\_OF\_MONTH, DATE, EXTRACT, FIRST\_DAY, FROM\_UTC\_TIMESTAMP, HOUR, HOURS\_BETWEEN, JULIAN\_DATE, MICROSECOND, MIDNIGHT\_SECONDS, MINUTE, MINUTES\_BETWEEN, MONTH, MONTHNAME, MONTHS\_BETWEEN, NEXT\_DAY, NEXT\_MONTH, NEXT\_QUARTER, NEXT\_WEEK, NEXT\_YEAR, QUARTER, ROUND, ROUND\_TIMESTAMP, SECOND, SECONDS\_BETWEEN, THIS\_MONTH, THIS\_QUARTER, THIS\_WEEK, THIS\_YEAR, TIME, TIMESTAMP, TIMESTAMP\_FORMAT, TIMESTAMP\_ISO, TIMESTAMPDIFF, TIMEZONE, TO\_CHAR, VARCHAR\_FORMAT, WEEK, WEEK\_ISO, WEEKS\_BETWEEN, YEAR, YEARS\_BETWEEN, YMD\_BETWEEN.

If a program uses functions that are not listed here, for example user defined functions, then you can notify the Compiler about these functions with the new configuration property:

```
iscobol.compiler.db2.fun.<function-name> = <sql type>
```

You can have multiple occurrences of this property, one for each function.

For example, to configure a user defined function named MY\_FUNC\_DATE that receives a date as parameter, you can add this entry to the Compiler configuration:

```
iscobol.compiler.db2.fun.my_func_date=DATE
```

Where "DATE" is the constant in java.sql.Types according to the Java documentation:

<https://docs.oracle.com/javase/8/docs/api/java/sql/Types.htm>.

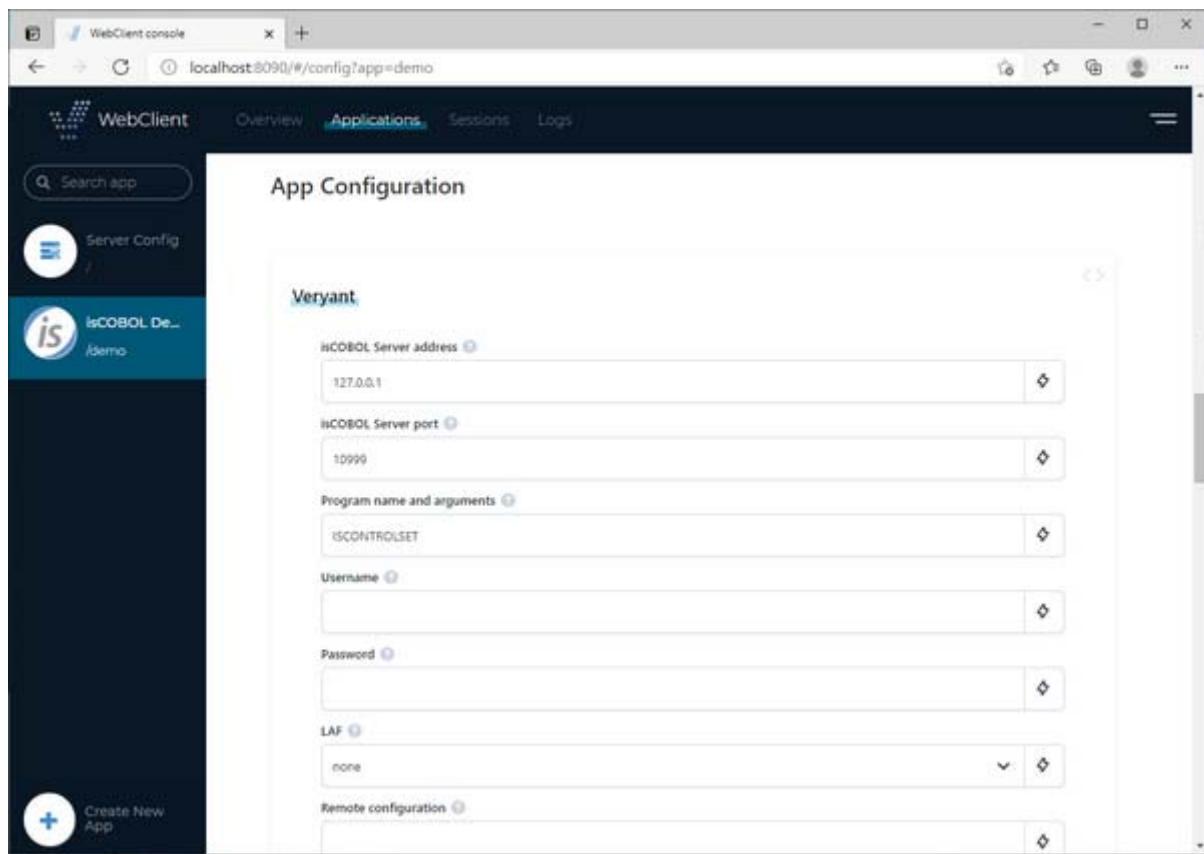
## WebClient

isCOBOL WebClient has been removed from the isCOBOL EIS suite of products, and is now a separate product that can be purchased as an add-on to the runtime system.

The isCOBOL Evolve 2021 R1 release of WebClient includes many new features and capabilities not available in previous versions:

- Support for Java 11  
isCOBOL WebClient now supports both Java 8 and Java 11, while previous releases only supported Java 8. OpenJDK is also qualified to be used.
- Better handling of mobile devices, especially those with touch screens
- Support for Hi-DPI (Retina) display
- Accessibility support based on WAI-ARIA 1.2 standard
- New layout for the dashboard: the dashboard has been redesigned for better legibility and usability. For example, in the list of sessions, active and finished sessions are no longer merged together; they're listed in separate pages. The app configuration page has been improved, with the configuration fields logically grouped in thematic areas, as depicted in Figure 8 - App configuration.

Figure 8 - App configuration



- Better handling of mobile devices, especially those with touch screens. As depicted in Figure 9 – WebClient Mobile bar, a new status bar is available for common screen's operation like copy and paste.

Figure 9 - WebClient Mobile Bar



- Separate admin application

The admin application is now a separate application, running on a dedicated port and can manage multiple WebClient servers at once, a useful feature when deploying in a clustered environment.

By default, the WebClient server starts on port 8080 and the WebClient admin starts on port 8090. These default ports can be changed by editing the jetty.properties configuration files installed along with the product.

The multiple WebClient servers managed by the WebClient admin could reside either on the same server (listening on separate ports) or on separate servers. In order to manage multiple WebClient servers you

must specify their WebSocket URL in the `webclient/admin/webclient-admin.properties` configuration file, e.g.

```
webclient.server.websocketUrl = ws://localhost:8080,ws://server2:8080
```

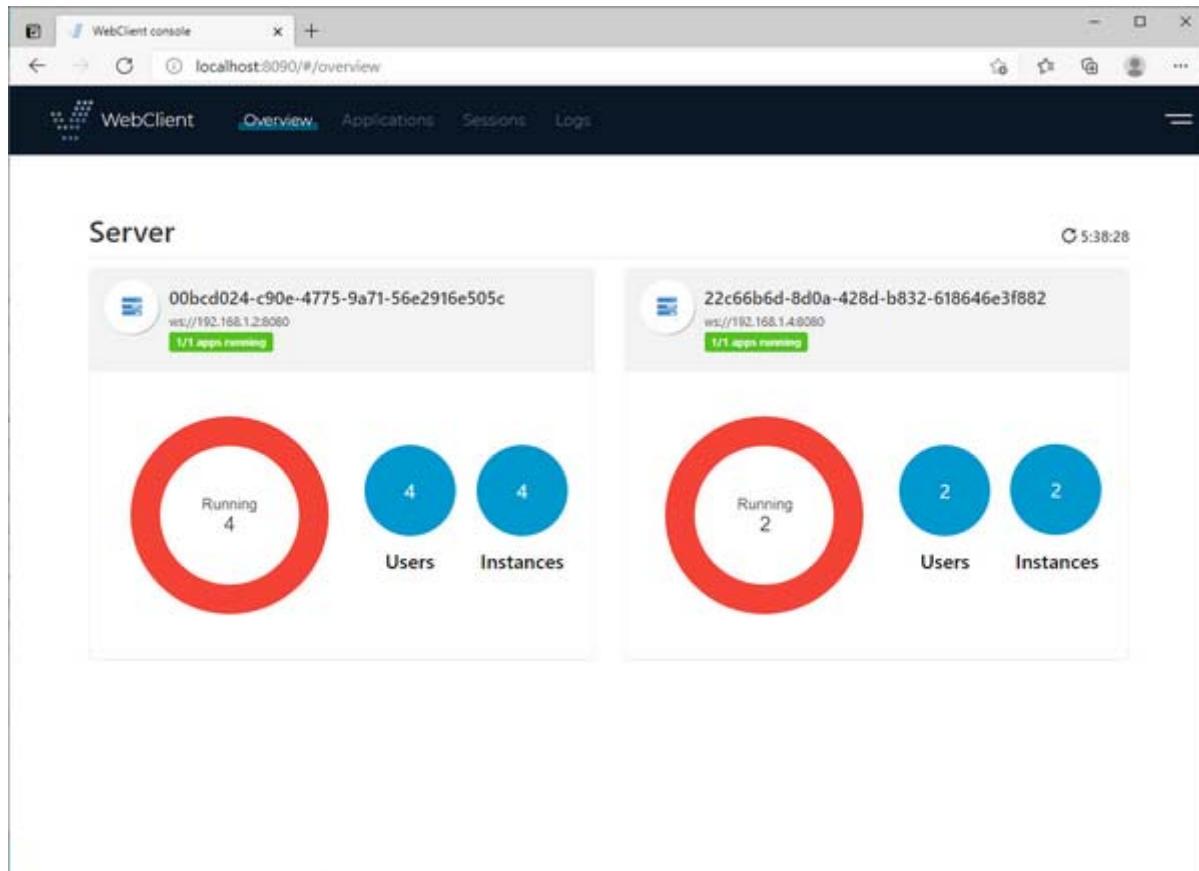
To enhance security, the admin application and the WebClient server must share the same “secret key” in order to establish a connection. Edit the `webclient/admin/webclient-admin.properties` configuration file to set the secret key for the admin app. Edit the `webclient/webclient.properties` configuration file to set the secret key for a WebClient instance.

Having the WebClient admin separated from the WebClient servers is particularly useful in clustered Cloud or distributed environments to improve the scalability of the software. As an example, in an e-commerce application whose workload increases in specific periods of the year (i.e. Black Friday and Christmas) might require new servers to be added only in peak periods, to better support the increased number of requests.

All the servers can now be monitored and managed from your own PC having the WebClient admin running locally. Separating the admin console from the runtime section results in increased security, since the admin console does not need to be installed in production servers.

The Figure 10, Monitor two WebClient servers, shows how two WebClient servers appear in the admin application. In this instance, there are four sessions running on the first server and two sessions running on the second server.

Figure 10.- Monitor two WebClient servers



- Multilanguage interface

It's now possible to choose the language that WebClient uses in the user interface.

Changing the language affects all the messages generated by WebClient (i.e. “Your network connection is slow”) as well as WebClient screens (i.e. the login screen).

Several languages are provided along with the product, and additional languages can be added by installing a dedicated msg.json file in the WebClient’s “lang” folder.

Figure 11, Language change, shows how to switch the language in the home page of WebClient.

Figure 11. Language change

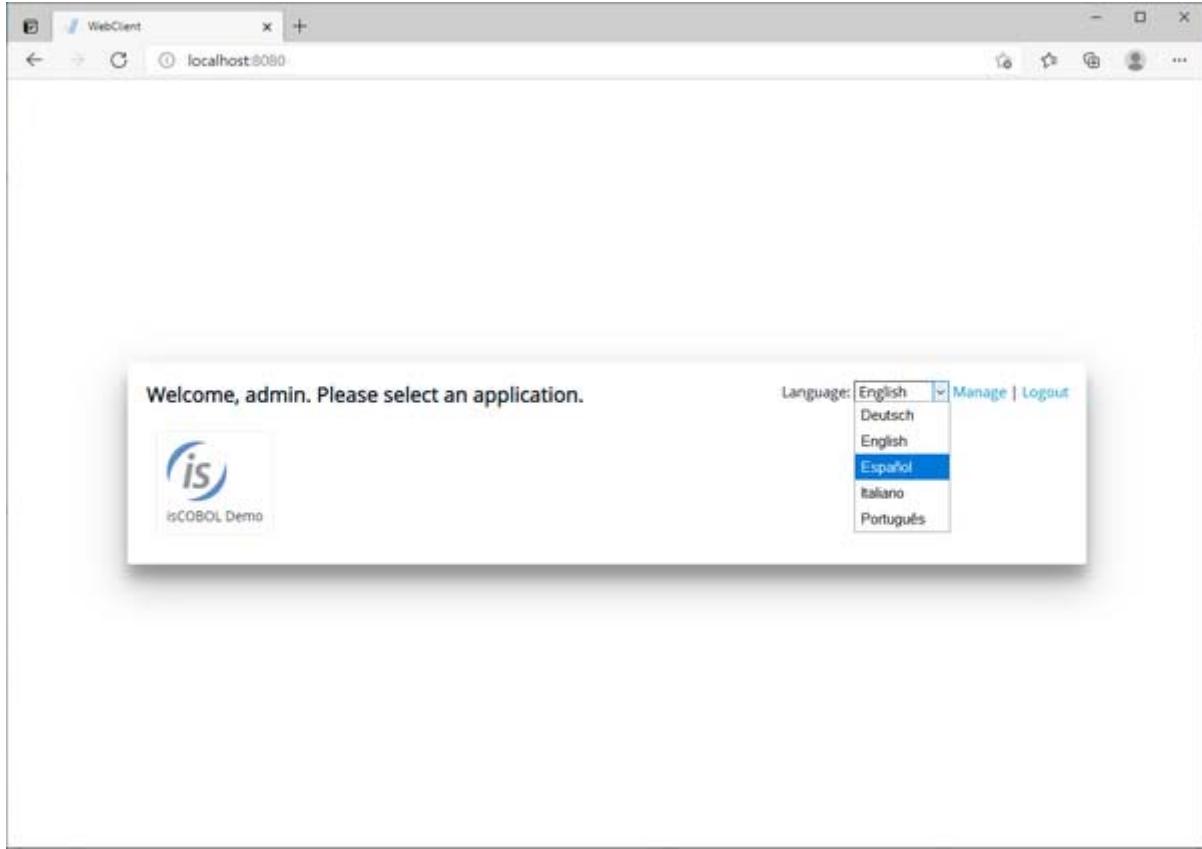
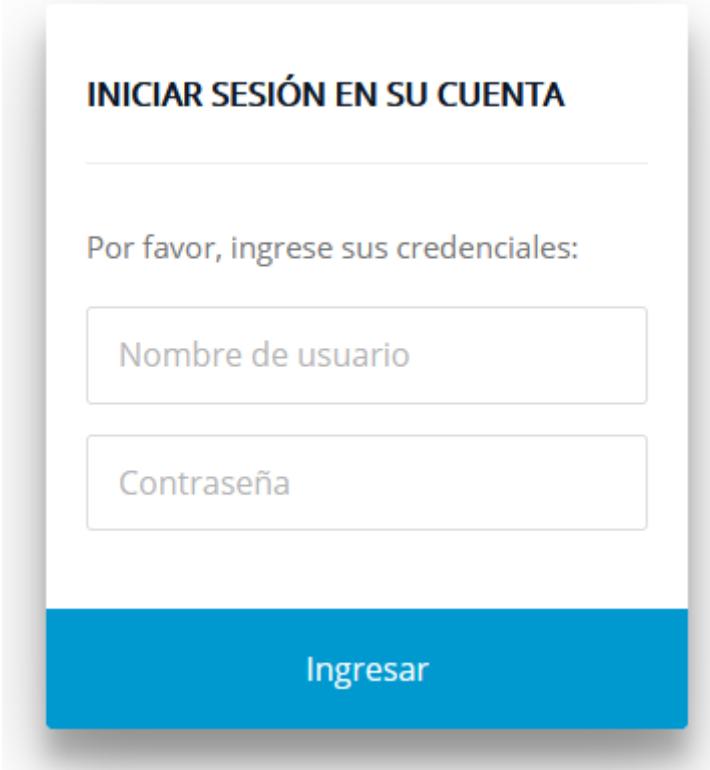


Figure 12 – Spanish login, shows login screen after Spanish has been selected.

Figure 12 - Spanish login



- Tomcat compliant

Even though WebClient comes with an embedded Jetty server, it is also possible to deploy it in an external servlet container like Tomcat. Other J2EE servers can work as well, as long as they support the Servlet 3.0 spec.

To deploy WebClient to Tomcat follow these steps:

- a. Make a copy of webclient-server.war from the isCOBOL SDK to Tomcat's webapps folder
- b. In webclient.properties set the property webclient.server.websocketUrl to

```
ws://localhost:<port>
```

with the port that Tomcat is running on

- c. In catalina.properties add the following properties:

```
webclient.warLocation=webapps/webclient-server.war
webclient.configFile=<path to WebClient's webclient.config file>
webclient.tempDirBase=<path to WebClient's tmp folder>
webclient.rootDir=<path to WebClient root>
```

Tomcat should be executed from its root folder, otherwise the path of webclient.warLocation needs to be adjusted.

# GUI enhancements

Many improvements to GUI controls have been implemented in this release. The tabcontrol has been enhanced allowing customization of colors and borders. Check-box, Push-buttons and Radio-buttons can now have rollover and disabled colors. All the container controls can now use a background image or gradient colors. Additional properties have been added in the supported GUI control syntax to allow a richer user interface.

## Tab-control enhancements

The isCOBOL compiler now supports new properties in a tab-control, allowing more customization. This is the list of new properties supported by this control:

- ACTIVE-TAB-BORDER-COLOR sets the border color of the active tab.
- ACTIVE-TAB-BORDER-WIDTH sets the width of the four borders of the active tab
- ACTIVE-TAB-COLOR, ACTIVE-TAB-BACKGROUND-COLOR and ACTIVE-TABFOREGROUND-COLOR set the color of the active tab's title page
- TAB-BORDER-COLOR sets the border color of tabs
- TAB-BORDER-WIDTH sets the border widths of every tab in a Tab-Control control (top, left, bottom and right)
- TAB-WIDTHS is useful to set the size of each tab's title page

The new NO-BOX style is useful to completely remove the tab borders.

In addition, existing properties previously supported only on Accordion tab-controls are now supported on all tab-control types:

TAB-COLOR, TAB-BACKGROUND-COLOR and TAB-FOREGROUND-COLOR set the color of the page titles.

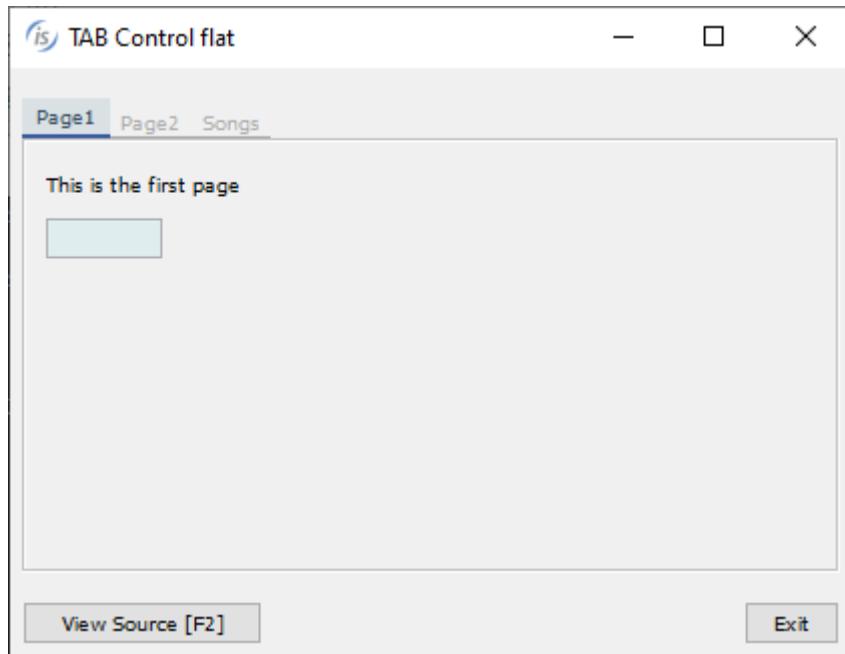
TAB-FLAT sets the flat style for pages.

An example of a flat tab-control with customized colors and borders is shown below:

```
03 tb1-container tab-control
  line 2 col 2 lines 17 cells size 68 cells
  allow-container tab-flat
  active-tab-border-width (0 0 3 0)
  tab-border-width (0 0 2 0)
  tab-foreground-color rgb x#ACACAC
  tab-border-color rgb x#ACACAC
  active-tab-border-color rgb x#395a9d
  active-tab-background-color rgb x#dae1e5
  active-tab-foreground-color rgb x#354c5c
.
```

Figure 13, Flat Tab-control, shows a program running with the tab-control containing the control defined in the code snippet. The active page is marked with a colored underlined border, typical of modern web applications.

Figure 13. Flat Tab-control



### Rollover and Disabled colors

The check-box, push-button and radio-button controls can now have specific rollover and disabled colors: when the cursor pointer is over a control the rollover color is applied, when the control is disabled the disabled color is applied.

The new properties to set the rollover color are: ROLLOVER-COLOR to set the COBOL color for both background and foreground, or ROLLOVER-BACKGROUND-COLOR and ROLLOVER-foreground-COLOR to set the background and foreground colors separately.

The properties to set the disabled color are: DISABLED-COLOR and DISABLEDBACKGROUND-COLOR, DISABLED-foreground-COLOR.

The following code snippet declares a push-button with the new color properties:

```
03 pb1 push-button
  line 8 col 3 lines 2 size 15 cells
  title "Button" self-act
  enabled e-controls
  flat
  background-color rgb x#C5DCEA
  foreground-color rgb x#354C5C
  Disabled-Background-Color rgb x#DDE4EF
  Disabled-Foreground-Color rgb x#4B6C83
  Rollover-Background-Color rgb x#A6F9DE
  Rollover-Foreground-Color rgb x#D51515
.
```

Figure 14, Rollover colors and Figure 15, Disabled colors show the new properties in action.

In Figure 14 the mouse is over the push-button labelled "Button", and in Figure 15 all controls are disabled.

Figure 14. Rollover colors

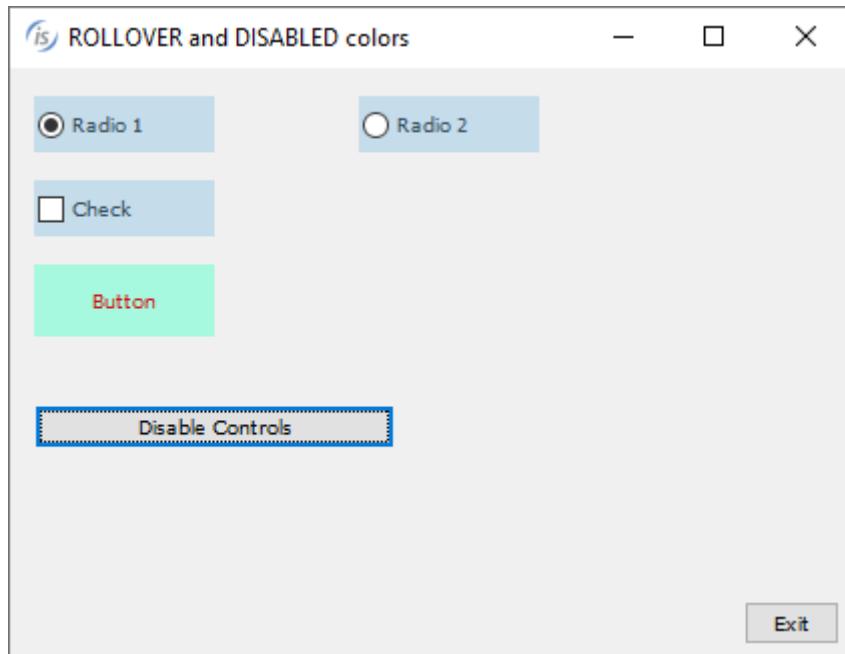
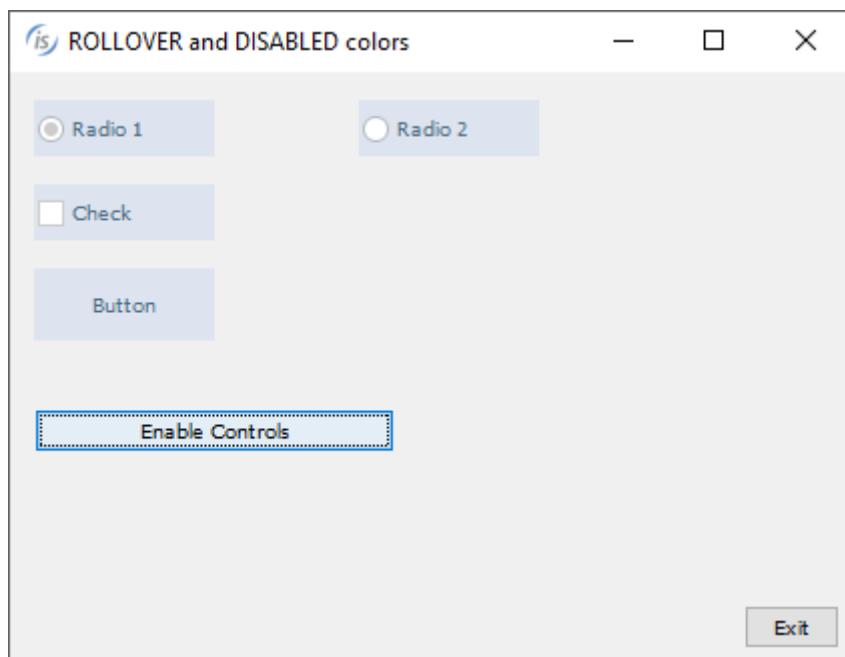


Figure 15. Disabled colors



### Background image and gradient on containers

Container controls like frame, list-box, ribbon, scroll-pane, tab-control, tree-view, tool-bar and window can now have a background image. This can help in modernizing COBOL applications.

The properties to set the background image are: BACKGROUND-BITMAP-HANDLE to set the handle of a loaded image and BACKGROUND-BITMAP-SCALE to specify the image scaling rule to apply. The supported values are the same as the existing property BITMAPSCALE:

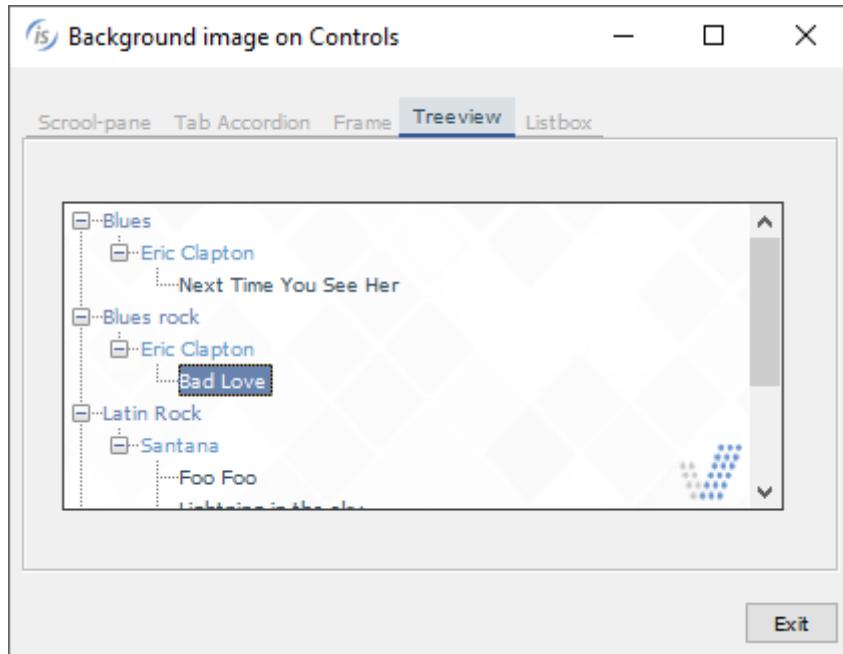
- 0 (default) = to maintain the original image without altering it. The image is cropped if larger than the available space, or is aligned in the top-left corner if it's smaller
- 1 = to resize the image automatically to fit completely the control area. The aspect ratio may not be preserved with this option
- 2 = to resize the image, keeping the aspect ratio. The image may not fill the available space completely, if the aspect ratio of the control is not the same as the image

The following code snippet declares a tree-view with a background image with rule of scale 1:

```
05 Tv1 tree-view
  line 3 col 4 lines 11 size 60
  buttons lines-at-root show-sel-always
  selection-background-color rgb x#6883AE
  selection-foreground-color rgb x#FFFFFF
  background-bitmap-handle hWatermark
  background-bitmap-scale 1.
```

Figure 16, Background image on container controls, shows the result of code-snipped of tree-view definition.

Figure 16. Background image on container controls



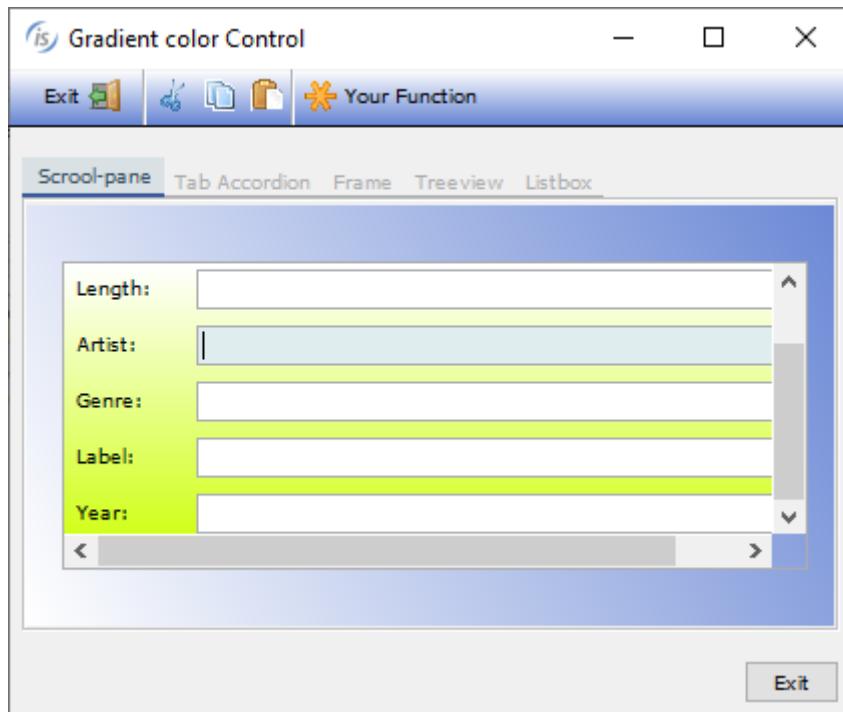
The same list of “container” controls also support gradient colors, while in previous releases gradients were supported only on a window.

The following code snippet shows a portion of a screen section where a tab-control and a scroll-pane are declared. There is also a procedure division snippet code that shows the creation of a tool-bar control:

```
03 tb1-container tab-control
    line 2 col 2 lines 17 cells size 68 cells
    ...
    gradient-color-1 w-gradient-color-blue-1
    gradient-color-2 w-gradient-color-blue-2
    gradient-orientation gradient-northeast-to-southwest.
03 tb1-container-page1 tab-group Tb1-container tab-group-value 1.
05 scroll-pane-1 scroll-pane
    line 3 column 4 size 62 lines 11
    ...
    gradient-color-1 w-gradient-color-yellow-1
    gradient-color-2 w-gradient-color-yellow-2.
    ...
display tool-bar control font control-font
    gradient-color-1 w-gradient-color-blue-1
    gradient-color-2 w-gradient-color-blue-2
    handle hToolBar.
```

Figure 17, Gradient on container controls, shows the gradient properties in action on the tool-bar, tab-control and scroll-pane defined in the previous code snippet.

Figure 17. Gradient on container controls



The new control options can easily be applied to existing programs using the “code injection” compiler feature, making modernization of existing programs as easy as adding a compiler configuration and recompiling.

## Additional GUI enhancements

The entry-field control supports a new property named TEXT-WRAPPING to set the rule for multiline wrapping; allowed values are:

- 0 AUTO-WRAP (default) implements CHAR-WRAP for entry-fields with national value set and WORD-WRAP otherwise
- 1 WORD-WRAP
- 2 CHAR-WRAP

Following is an example of declaring an entry-field with char-wrap:

```
03 ef1 entry-field  
  line 2 col 2 size 40 lines 10  
  text-wrapping 2.
```

The tree-view control supports a new property, search-panel, to show a filtering field over the tree-view. For coherence, the grid supports the same property with same values:

-1: the search panel never appears on top of the control even if the user presses Ctrl-F

0: the search panel appears on top of the control when the user presses Ctrl-F. This is the default behavior for tree-view and grid

1: the search panel is always visible on top of the control. The user can't dismiss it

The following code modifies a tree-view, making the search-panel always visible:

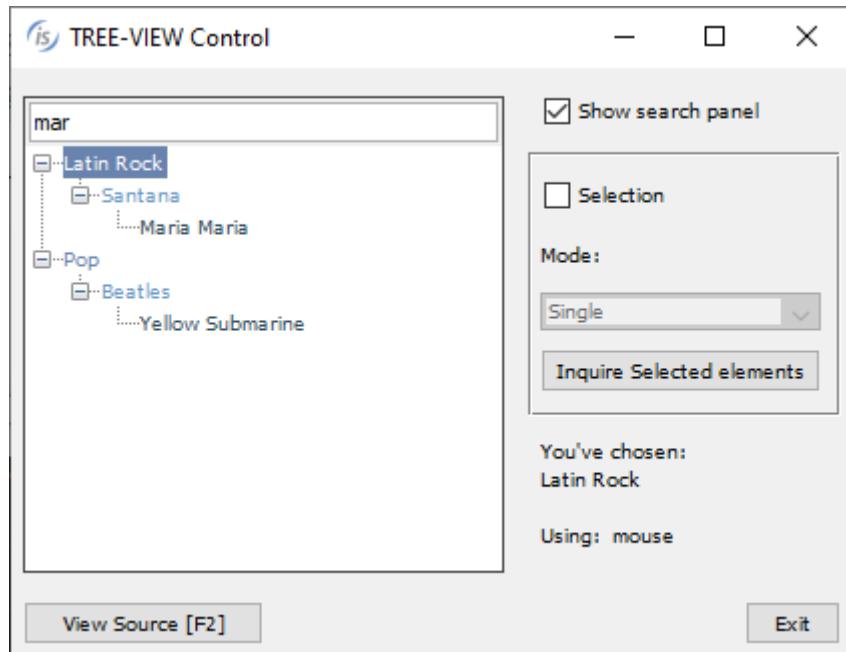
```
modify Tv1 search-panel 1
```

The search panel is triggered by default when the user presses Ctrl+F, but it can be customized on a different key combination, for example Ctrl+G by setting the configuration option

```
iscobol.key.*g=search=grid,print-preview,web-browser,tree-view
```

Figure 18, Tree-view search-panel, shows the items filtered after the user types “mar” as a filter. The search is case insensitive and the matching items will be rendered along with their parent items.

Figure 18. Tree-view search panel

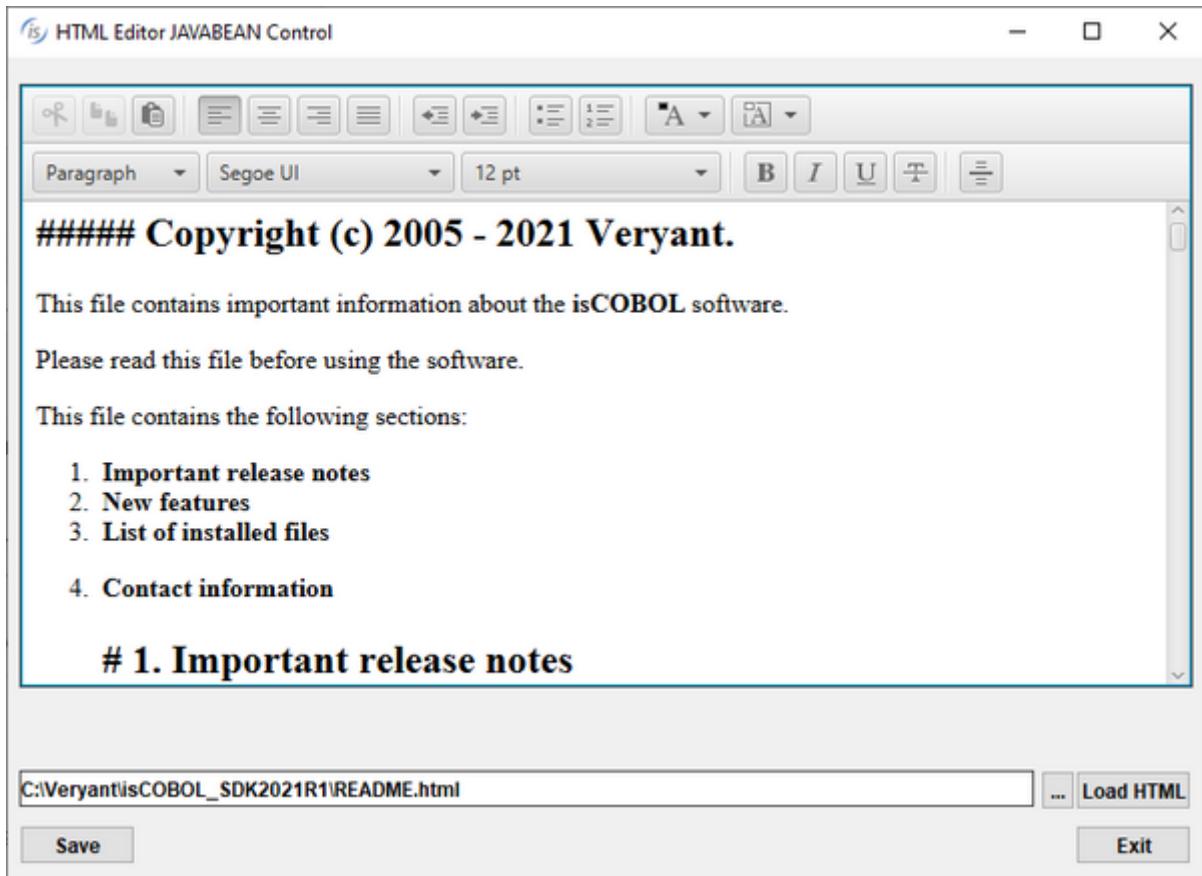


The java-bean control can now contain a JavaFX component, increasing GUI choices. The example below shows how add a java-bean embedding an HTML editor JavaFX component:

```
03 jb-editor java-bean
  line 2, col 2, lines 20, size 78
  clsid "javafx.scene.web.HTMLEditor"
  object my-editor.
```

Figure 19, FX HTMLEditor in Java-bean, shows the HTMLEditor that appears in the COBOL window at runtime.

Figure 19. FX HTMLEditor in Java-bean



The date-entry control supports a new property, ILLEGAL-DATE-VALUE, to set the value returned in case an illegal date is entered, and is used as shown below:

```
03 de1 date-entry
line 2, col 2, value-format DAVF-YYYYMMDD
illegal-date-value 99991231.
```

In addition, the BORDER-COLOR and BORDER-WIDTH properties previously supported in Entry-fields are now supported on Date-entry and Push-button controls, resulting in richer border customization.

The C\$OPENSAVEBOX routine will now use native dialogs when run on Microsoft Windows. The previous LAF-dependent user interface is still available when running isrun or isclient under Linux or Mac systems or under any web browser via WebClient.

## isCOBOL Compiler

Starting from isCOBOL Evolve 2021R1, the compiler supports Lambda expressions through the new operator "->".

SORT statements on occurs containing dynamic data items are now fully supported, as well as nested dynamic capacity tables in multilevel occurs.

Additional syntax is now supported to improve compatibility with other COBOLs like MicroFocus and RM/COBOL.

## Lambda expression

A lambda expression, also known as anonymous function, is a block of code that can be passed as an argument to a function call. This is supported in many languages like C# and Java, and now is supported in isCOBOL as well. This is useful when using OOP (Object Oriented Programming) to invoke existing java classes, and when writing OOP CLASS-ID directly in COBOL. The required operator in the COBOL source is -> and it needs to be written before the class name. Lambda expressions are similar to methods, but they do not need a name, and can be implemented right in the body of a method.

For example, the following CLASS-ID myclass lists all the \*.cbl files in the current directory by printing their name on the system output. Files that don't have a cbl extension are not listed. The filtering is performed by the filterFileName() method that matches the accept() method in the FilenameFilter interface and is invoked via Lambda, see the second statement in the procedure division of the main() method.

This is the full source of the CLASS-ID:

```

identification division.
class-id. myclass as "myclass".
configuration section.
repository.
class j-string as "java.lang.String"
class j-string-arr as "java.lang.String[]"
class j-io-file as "java.io.File"
class j-system as "java.lang.System"
.

identification division.
factory.
procedure division.
identification division.
method-id. main as "main".
working-storage section.
77 curr-dir object reference j-io-file.
77 file-list object reference j-string-arr.
77 len int.
77 i int.
linkage section.
01 args object reference j-string-arr.
procedure division using args.
main.
  set curr-dir to j-io-file:>new(".").
  set file-list to curr-dir:>list(>myclass:>filterFileName).
  set len to file-list:>length.
  perform varying i from 0 by 1 until i >= len
    display file-list(i)
  end-perform
  goback.
end method.
identification division.
method-id. filterFileName as "filterFileName".
working-storage section.
77 ret object reference "boolean".
linkage section.
01 f object reference j-io-file.
01 n object reference j-string.
procedure division using f, n returning ret.
main.
  if n:>toLowerCase:>endsWith(".cbl")
    set ret to true
  else
    set ret to false
  end-if
  goback.
end method.
end factory.
end class.

```

## SORT with dynamic tables

The SORT statement was used in previous releases to SORT data inside fixed length groups. In the isCOBOL Evolve 2021 R1 release it's also possible to sort "group-dynamic" groups, which are groups containing dynamic length data item, such as "ANY LENGTH" items:

```
01 w-table-contacts.  
 05 w-contacts occurs 9 times.  
    10 w-contact-cod pic 9(3).  
    10 w-contact-name pic x(50).  
    10 w-contact-notes pic x any length.
```

The following statements:

```
sort w-contacts on ascending key w-contact-name  
sort w-contacts on ascending key w-contact-notes
```

can now be compiled without any warnings "Dynamic items will be ignored: WCONTACTS" and when running the table will be sorted correctly.

Nested OCCURS DYNAMIC data structures are also now supported, such as:

```
01 w-table-contacts.  
 05 w-company-occ occurs dynamic capacity cap-company.  
    10 w-company-cod pic 9(6).  
    10 w-company-name pic x(50).  
    10 w-company-contacts.  
      15 w-contact-occ occurs dynamic capacity cap-contact.  
        20 w-contact-cod pic 9(3).  
        20 w-contact-name pic x(50).  
        20 w-contact-notes pic x any length.
```

allowing the use of a SORT statement such as:

```
sort w-company-occ on ascending key w-company-name
```

No Warning or Severe Error will be issued when compiling, all company names will be sorted according to the key specified on the statement, and all the nested occurs dataitems will be moved to their correspondent parents.

## Improved Compatibility with other COBOLs

To improve the compatibility with the MicroFocus COBOL dialect, and to offer additional syntax to isCOBOL users without the need to use any compiler option, isCOBOL compiler now supports the following:

- LOCAL-STORAGE SECTION is now supported for METHOD-IDs. In previous releases all the data items declared inside WORKING-STORAGE of a METHOD ID were completely local. Now, if the LOCAL-STORAGE SECTION is declared the WORKING-STORAGE SECTION becomes shared, and on

subsequent executions of the same method the variables under WORKING will retain previous values, while the variables under LOCAL will always be initialized to its original values. Following is a code snippet that shows the declaration of LOCAL-STORAGE in a METHOD-ID:

```
identification division.  
method-id. methodCompute as "methodCompute".  
working-storage section.  
77 w-var-1 pic 9(3) value 0.  
local-storage section.  
77 l-loc-1 pic 9(3) value 0.  
procedure division.
```

- OBJECT REFERENCE declared on main level, 01 or 77, can now have the OCCURS clause, and the code below

```
01 obj-occ occurs 5 object reference.  
77 jint-occ object reference jint occurs 9.  
77 jstr-occ object reference "java.lang.String" occurs 20.
```

can now be compiled. These occurs use indexes that starts from 1, following the COBOL rule instead of using Java indexes [ ] that start from 0.

- Concatenation using figurative constants is now fully supported, as shown in the following code:

```
77 w-desc1 pic x(10) value "ab" & low-value.  
77 w-desc2 pic x(10) value x"3132" & zero.  
...  
move "xyz" & high-value to w-desc3
```

- Data description entry not terminated by a dot is now supported returning a compiler Error but not a Severe Error, and it can be suppressed with the compiler configuration:

```
iscobol.compiler.messagelevel.188=0
```

- The code snippet:

```
01 VAR-GROUP  
03 VAR1 PIC X  
03 VAR2 PIC X  
77 VAR77 PIC X  
78 CONST78 VALUE "SALC"
```

is supported and does not require that you manually add the missing dots, since they are assumed automatically by compiler.

- The SET statements to assign pointers are now more flexible, allowing the OF clause to be optional, making the following code

```
set ptr1 to address wrk1  
set address lk1 to ptr1  
set address lk2 to address wrk1
```

equal to

```
set ptr1 to address of wrk1
set address of lk1 to ptr1
set address of lk2 to address of wrk1
```

To improve compatibility with RM/COBOL, isCOBOL compiler has improved the -cr option to accept the popup window RM syntax uses to create and remove a popup window on the DISPLAY statement with the CONTROL clause. Now there is no need to manually change the COBOL source during RM migrations. A code snippet such as:

```
01 WINDOW-CONTROL-BLOCK.
03 WCB-HANDLE PIC 999 BINARY(2) VALUE 0.
03 WCB-NUM-ROWS PIC 999 BINARY(2).
03 WCB-NUM-COLS PIC 999 BINARY(2).
...
03 WCB-TITLE PIC X(40).
```

declares a window definition, and the following DISPLAY statement creates and removes the pop-up window:

```
MOVE 10 TO WCB-NUM-ROWS
MOVE 40 TO WCB-NUM-COLS
MOVE "Customer list" TO WCB-TITLE
DISPLAY WINDOW-CONTROL-BLOCK LINE 5 POSITION 20
CONTROL "WINDOW-CREATE, REVERSE"
DISPLAY WINDOW-CONTROL-BLOCK CONTROL "WINDOW-REMOVE"
```

To simplify migrating from RM /COBOL, a new File Connector has been created in the isCOBOL Evolve 2021R1 release to allow full access to existing RM/COBOL indexed files.

This is currently available on both 32 and 64-bits Windows environments.

The isCOBOL configuration settings to use this connector are:

```
iscobol.file.index=rmc
# set the below property if rmc.exe is not located in PATH
iscobol.file.connector.program.rmc=/path/to/rmc.exe
```

The RMC connector can be used with any Veryant product and utility, and it can be used to access RM indexed files from the isCOBOL runtime, isCOBOL File Server, isCOBOL UDBC and utilities like GIFE and ISMIGRATE.

A file connector is also useful when running in a “mixed COBOL” environment, and data files need to be shared. For example, if an RM installation needs to quickly deploy a new portion of an application that needs to be executed in a web browser, this can be deployed using isCOBOL WebClient and can run concurrently with the original application still running under RM. No full migration is needed to provide added benefits for users, and migration can proceed in parallel with the implementation of new features.

## isCOBOL Runtime

isCOBOL Evolve 2021R1 improves runtime performance across the board, new runtime configurations have been added to customize the behavior when running applications, and now there is support for Oracle Tuxedo.

## Sort performance

Sort operations are common in COBOL applications, from interactive applications, such as reading from indexed or sequential files and outputting a sorted file to be presented to the user, to batch applications, where usually an external utility, ISSORT, or a call to the library routine C\$SORT are used to perform sorting operations. For both scenarios, the newest isCOBOL release has optimized the core of sort operations, and all COBOL programs will gain better performance as a result.

A table of performance gains is shown in Figure 20, Sort performances, comparing isCOBOL Evolve 2020R2 to isCOBOL Evolve 2021R1 using the same configuration:

```
iscobol.sort.memsize=33554432  
iscobol.sort.maxfile=8
```

The test was run in Windows 10 64-bit on an Intel Core i7 Processor 8550U+ clocked at 1.80 GHz with 16 GB of RAM, using Open-JDK 11.0.10. All times are in seconds, number of records involved in the sort: 1 million.

Figure 20. Sort performances

Operation:	2020 R2	2021 R1
statement SORT USING ... GIVING... INPUT SEQ OUTPUT SEQ	13.29	2.61
statement SORT USING ... GIVING... INPUT LIN SEQ OUTPUT LINE SEQ	13.57	2.32
statement SORT USING ... GIVING... INTPUT INDEX OUTPUT SEQ	19.11	8.97
statement SORT USING ... GIVING... INTPUT INDEX OUTPUT LINE SEQ	19.51	9.12
ISSORT utility INPUT SEQ	14.14	3.10
ISSORT utility INPUT INDEX	20.19	10.91
C\$SORT library routine INPUT SEQ	12.42	2.25
C\$SORT library routine INPUT IDX	18.16	10.66
Total	130.39	49.94

## New configurations

Starting from the isCOBOL Evolve 2021 R1 release, all the configuration properties are searched for in the external environment, with the convention of having dots replaced by underscores in the name. This simplifies the migration of existing applications that only use environment variables instead of relying on a configuration properties file. For example, running the following statement:

```
accept var from environment "myapp.var"
```

the value of the environment variable MYAPP\_VAR is returned.

The following is the list of new runtime configurations:

```
iscobol.apply_code_path=true
```

to apply code\_prefix to absolute paths used in CALL statements.

By default, the configuration option is set to false, and the code\_prefix is applied only to relative paths used in the CALL statement. As an example, a COBOL program that runs the following statement:

```
CALL "/myapp/cobol/listusr"
```

with the configuration options:

```
iscobol.apply_code_path=true  
iscobol.code_prefix=/opt/company
```

will cause the call program to load the LISTUSR.class file from the directory /opt/company/myapp/cobol.

The configuration option

```
iscobol.exception.dumpfile=value
```

is used to customize the name of the file generated when using the existing configuration  
iscobol.exception.message=3.

Special characters are supported in the value of the property, and act as a placeholder for the actual value:

- %p to identify the program name
- %d to identify the current date in the form YYYYMMDD
- %t to identify the current time in the form HHMMSSTTT
- %u to identify the username
- %h to identify the hostname

For example, if the following configuration is used:

```
iscobol.exception.dumpfile=/tmp/logs/%u/%d-%t-%p.dump  
iscobol.exception.message=3
```

the exception dump-file generated when user "user1" runs the program is named "20210420-115324678-PATCH56.dump" and it is created in the /tmp/logs/user1 folder.

A "+" character can be used at the start the configuration option to append to an existing file instead of overwriting it.

```
iscobol.extfh.keep_trailing_spaces=false
```

(default true) to remove trailing spaces in EXTFH interface. This is similar to other existing configurations for line sequential files, but this option is only used for the EXTFH interface.

The existing *iscobol.jdbc.timestampformat* now supports additional "S" values after the seconds "s", to include microseconds in ESQL timestamps instead of just the milliseconds. This increases timestamp precision when inserting or retrieving such fields from a database. For example:

```
iscobol.jdbc.timestampformat=yyyy-MM-dd-HH.mm.ss.SSSSSS
```

isCOBOL Evolve 2021 R1 implements a new routine, C\$NCALLRUN, to query the number of programs called using CALL RUN that are still running. This provides better monitoring of runtime environments where CALL RUN statements are used to run concurrent batch processing. For example, the code snippet

```
perform test after until w-count = 0
  call "c$ncallrun" giving w-count
  call "c$sleep" using 0.5
end-perform
```

shows how to wait for all threads generated by CALL RUN to terminate before exiting.

## Oracle Tuxedo integration

Tuxedo (Transactions for Unix, Extended for Distributed Operations) is a middleware platform used to manage distributed transaction processing in distributed computing environments. It distributes applications across multiple platforms, databases, and operating systems using message-based communications and distributed transaction processing.

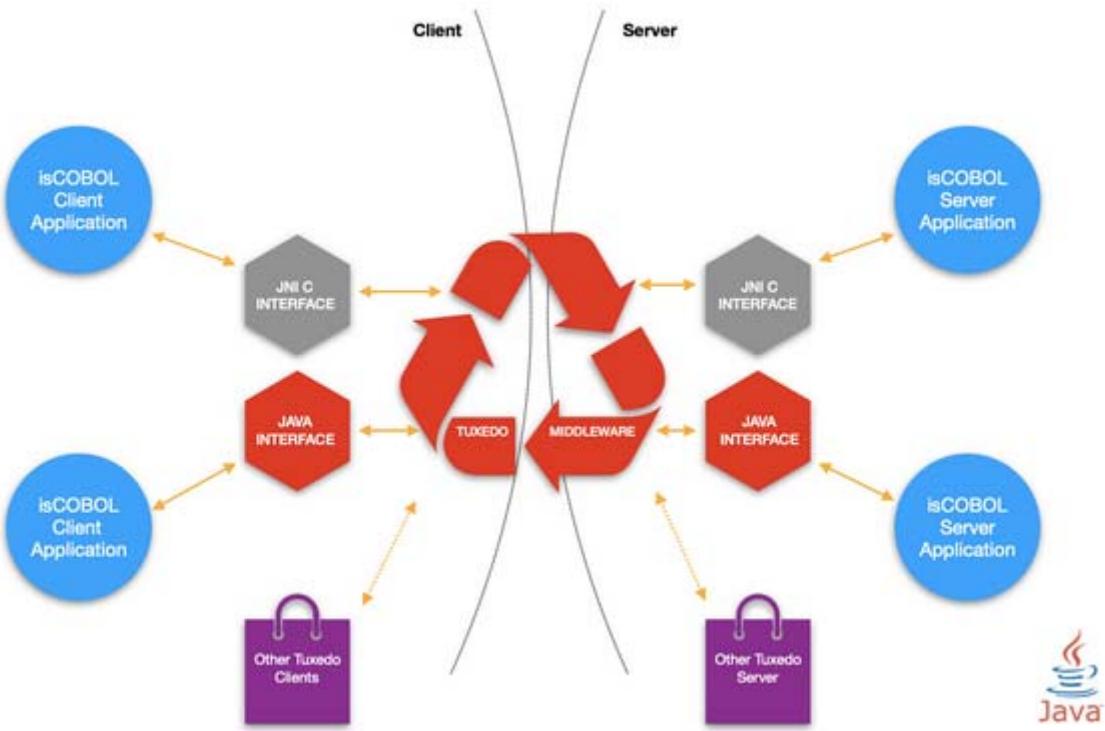
isCOBOL developers using the Oracle Tuxedo platform for distributed transaction processing and message-based application development can create Tuxedo clients and Tuxedo services from COBOL applications. The isCOBOL Evolve 2021R1 is qualified for Tuxedo 12 on all supported platforms.

isCOBOL and Tuxedo can work together in a distributed processing (client/server) environment providing two flavors of execution:

- Legacy COBOL approach, based on calls to the C routines provided by Tuxedo
- Java approach, based on OOP Java methods used to create clients and services

In a distributed processing environment, the interaction occurs as depicted in Figure 21, Tuxedo diagram, where isCOBOL can be used to develop both Client and Server applications using the JNI C interface, to migrate previous COBOL dialects used in Tuxedo or taking advantage of a Java interface to have a more optimized multithread model.

Figure 21. Tuxedo diagram



The newest release of the isCOBOL runtime includes improvements to C interoperability by providing:

- a new configuration `iscobol.shared_dlopen_null=false` (default true) to specify if functions called by the COBOL program are searched for in the current process.
- two new functions in the C API, `isCobolCallNoStop` and `isCobolCallNoStopEx`, are used to call isCOBOL from C without terminating the process if a STOP RUN statement is executed in the COBOL code.

These new C API functions and configurations are available in a C context and can be also be used by other applications. They are automatically used in the Tuxedo integration.

## isCOBOL Server

The routines available in the isCOBOL Application Server environment, that usually start with A\$, have been improved to provide communications between connected Thin Clients and to query the logon time. These features are integrated in the isCOBOL Panel and are useful in all architectures: ThinClient, WebClient or a mixed Thin + Web clients.

## ApplicationServer library routines

A new routine, A\$SEND\_MESSAGE, is now available to send a message to a thread ID running in an ApplicationServer environment without Multitasking. The syntax of the library routine is:

```
CALL "A$SEND_MESSAGE" USING TID,  
      msgText,  
      msgTitle
```

where the first parameter specifies the recipients' thread ID, the second specifies the text of the message and the third optional parameter specifies the title of the message box. By default, the message will appear as a standard graphical message box. This is usercustomizable by simply creating a program with the name A\$CUSTOM\_MESSAGE, and making it available in the client's CLASSPATH or code\_prefix.

For example, the following program will display received messages as a Notification window:

```
program-id. "A$CUSTOM_MESSAGE".  
working-storage section.  
77 n-win handle of window.  
linkage section.  
77 msgText pic x any length.  
77 msgTitle pic x any length.  
screen section.  
01 n-screen.  
  03 entry-field line 1, col 1  
    lines 10 cells, size 40 cells  
    no-box, multiline, read-only, value msgText.  
procedure division using msgText, msgTitle.  
main.  
  display notification window bottom right  
    lines 10 size 40 before time 500  
    visible 0 handle n-win  
  display n-screen upon n-win  
  modify n-win visible 1  
  goback.
```

The existing library routines A\$LIST-USERS and A\$CURRENT-USER now support an additional optional parameter that returns the user's login time.

```
call "A$LIST-USERS" using listusr-next usrlist  
      usr-id usr-name  
      usr-addr usr-pcname  
      usr-tid usr-prog  
      usr-type  
      usr-logon-time  
  
call "A$CURRENT-USER" using usr-id usr-name  
      usr-addr usr-pcname  
      th-id usr-prog  
      usr-type  
      usr-logon-time
```

where usr-logon-time is defined as:

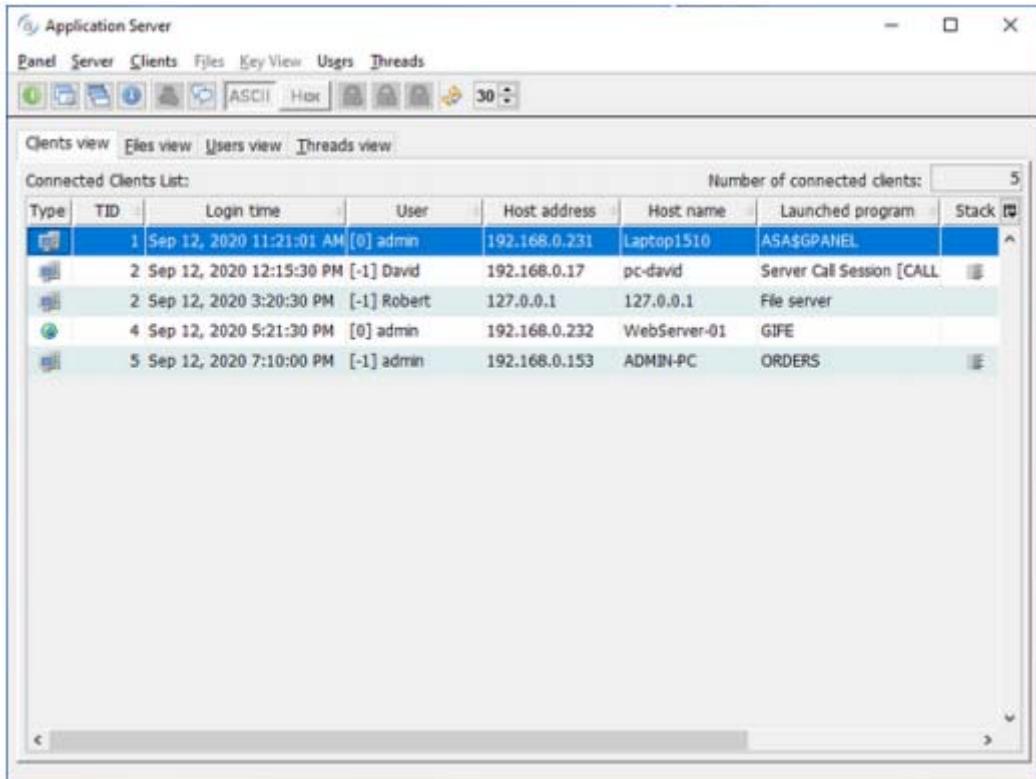
```
01 usr-logon-time pic x(16).
```

and contains the timestamp in YYYYMMDDHHNNSSCC format, where YYYY is the year, MM the month, DD the day, HH the hour, NN the minutes, SS the seconds and CC the hundreds of second. The time is returned in the UTC time zone.

## isCOBOL Panel

The isCOBOL Application Server Panel has been enhanced and uses the new features of A\$\* routines, providing an additional column “Login time” that helps in sorting the Client connections, for example to easily identify which ones are the oldest. Figure 22, isCOBOL Panel, shows all the clients of different types, including File Server and remote calls, all having the Login time set.

Figure 22. isCOBOL Panel



The message sending capability has been integrated in the isCOBOL Panel, allowing messages to be sent from the Panel to all or to a selected list of connected clients.

In Figure 23, Clients menu, the Administrator is choosing to Send a message to all clients, and in Figure 24, Send message window, the text has been input. After pressing the button Send, the message will be delivered to connected ThinClients and WebClients, advising users of downtime due to maintenance. Figure 25, Message received from ThinClient, shows the message received on the ThinClient processes.

Figure 23. Clients menu



Figure 24. Send Message window

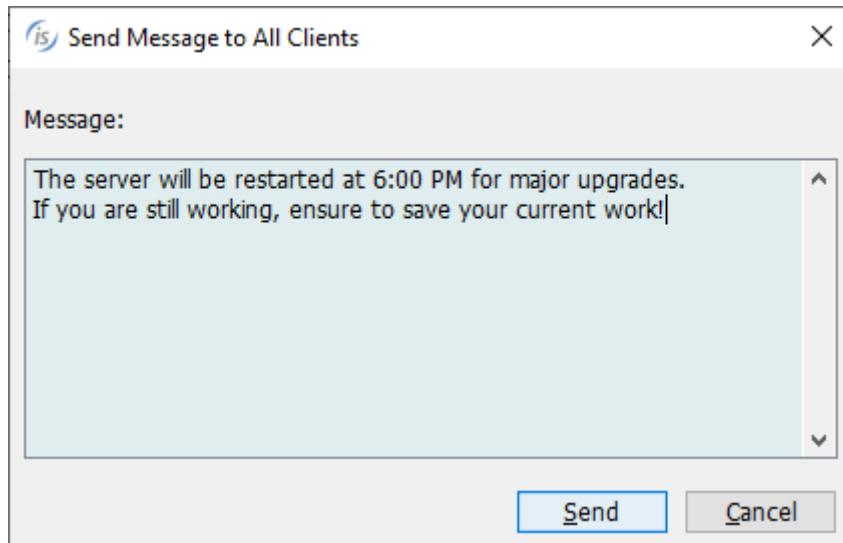
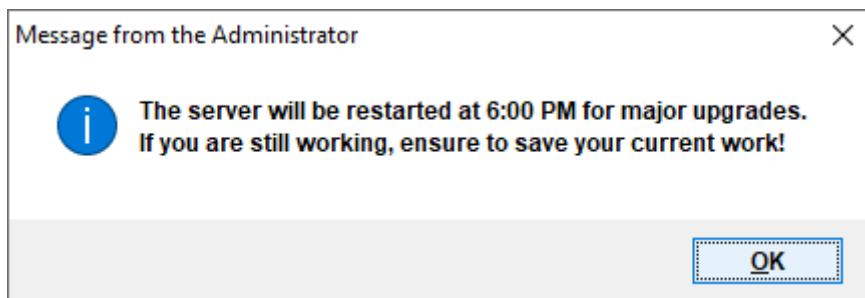


Figure 25. Message received from ThinClient



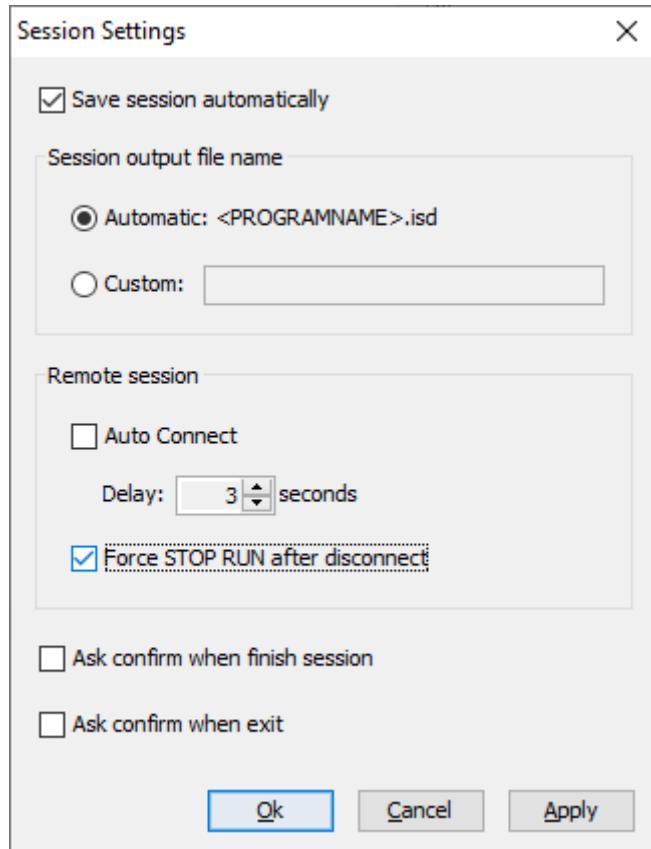
## isCOBOL Debugger

isCOBOL Remote Debugger is useful to debug processes that are running on a different computer, which is typical in architectures like isCOBOL Thin Client using the `isclient -d` option. Other applicable architectures such as Tomcat, WebClient and batch processes that are executed on a server without X11 support require the Debugger to be executed on a separate computer with a GUI environment.

In these scenarios the Remote Debugger is started by executing the "`iscrun -d -r IP port`" command.

The Remote Debugger has the ability to disconnect and reconnect, and with the latest release it has the option to issue a STOP RUN COBOL statement to terminate execution of the program after disconnecting. As shown in Figure 26, Debugger session settings, in order to activate this feature, simply check the new setting "Force STOP RUN after disconnect" in the Session Settings window.

Figure 26. Debugger session settings



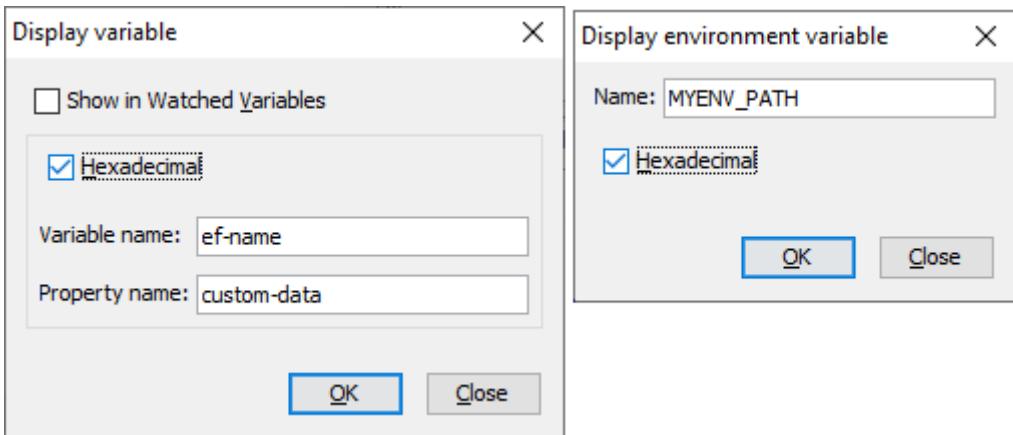
The debugger command "display" now supports -x option on properties and environment variables to show the value in hexadecimal for any piece of information inquired.

This is available in the debugger's command-line, using the commands:

```
display -x ef-name property custom-data  
display -x -env MYENV_PATH
```

as well as the graphical debugger window, as shown in Figure 27, Debugger display windows, in "Display variable" and "Display environment variable", checking the "Hexadecimal" box.

Figure 27. Debugger display windows



The isCOBOL Debugger has a new icon in the tool-bar, placed next to the combo box containing the source file name to identify whether sources are loaded from disk or extracted from a class.

By default, programs compiled with release 2020 R2 or greater will extract the source code from compiled classes, while programs compiled with previous releases will load source files from disk. Loading sources from disk can be forced by using a new debug configuration option:

```
iscobol.debug.embedded_source=false
```

This is useful, for example, when debugging preprocessed COBOL sources.

The new source location icon helps understand how the source code has been loaded, especially in mixed scenarios where programs are compiled with different compiler releases.

## isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. In isCOBOL Evolve 2021R1, the `HTTPClient` class can consume web services using the `PATCH` and `DELETE` methods, passing data in the request body.

### HTTPClient

`HTTPClient` is a class that allows COBOL programs to interact with Web Services. It has been updated to manage `DELETE` requests with data in the request body and `PATCH` requests.

The new method signatures are shown below:

```
public doPatch ( strUrl )
public doPatch ( strUrl, params )
public doPatchEx ( strUrl, content )
public doPatchEx ( strUrl, type, content )
public doPatchEx ( strUrl, type, content, hasDummyRoot )
public doDeleteEx ( strUrl, content )
public doDeleteEx ( strUrl, type, content )
public doDeleteEx ( strUrl, type, content, hasDummyRoot )
```

This provides a more comprehensive coverage of existing REST APIs than ever before.

## Additional improvements

All of the Veryant setups have been improved. A new c-tree RTG v3 release is included in isCOBOL Evolve 2021R1 and several utilities have been upgraded.

### Veryant setups

Two new graphical setups are available in the appropriate format to install isCOBOL Evolve: for Mac OS in .dmg format, and for the Linux operating system in .sh shell command.

The isCOBOL Evolve setup now includes both IDE and SDK products, making it easier to install products for developers that rely on the Eclipse-based IDE environment or for those who rely on command line environments using the tools of their choice.

Figure 28, New Mac setup, and Figure 29, New Linux setup, show the new look of the graphical installers.

Figure 28. New Mac setup

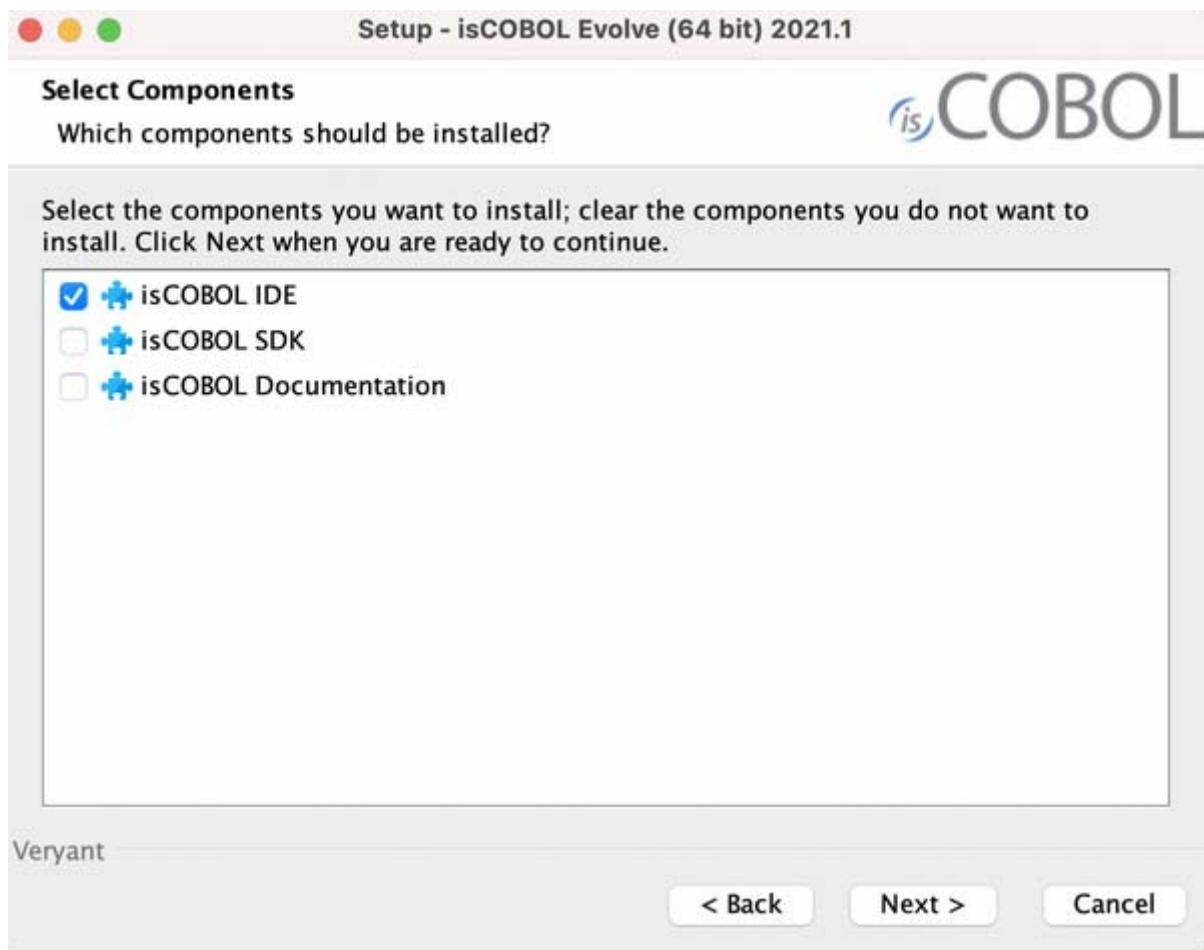
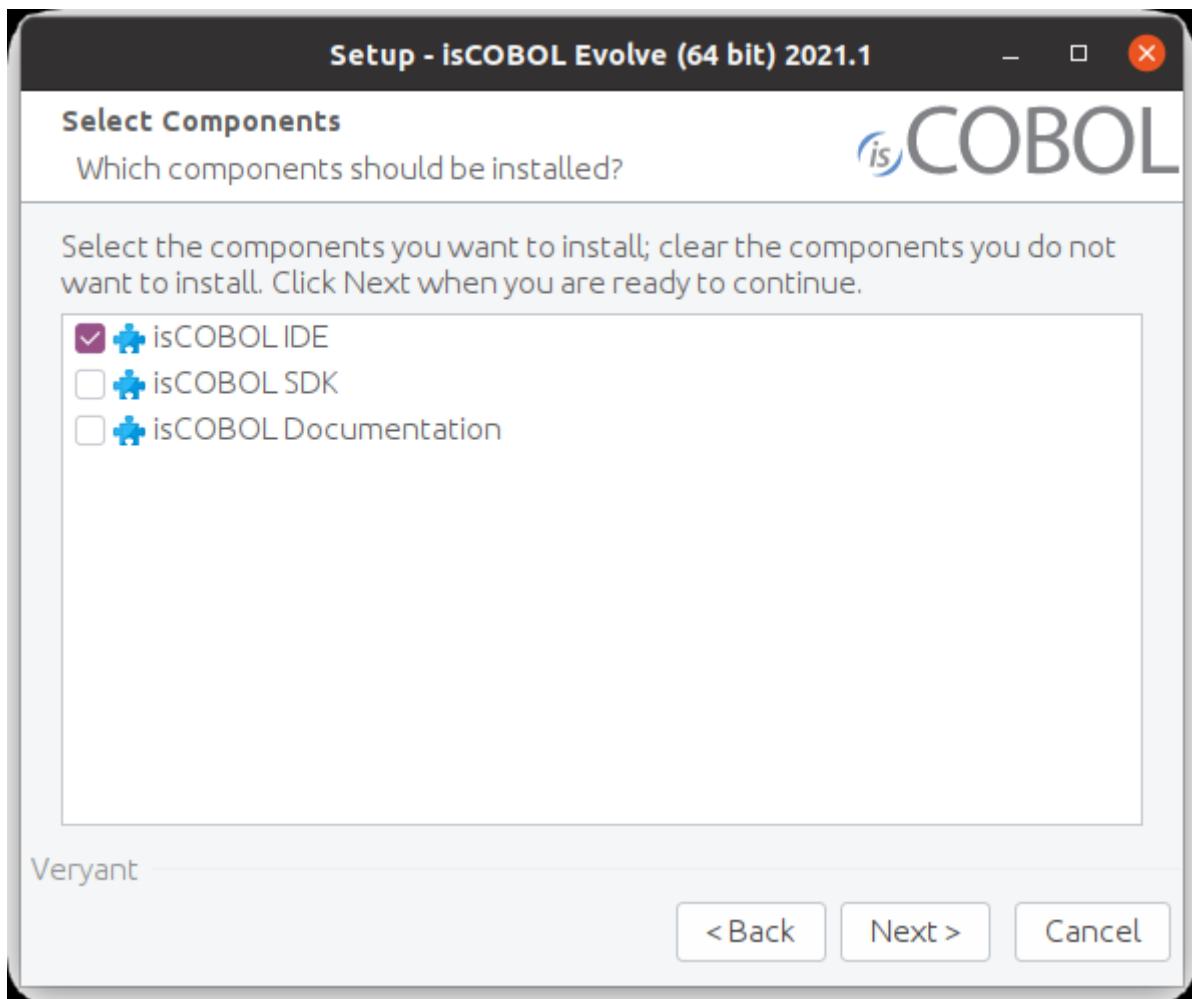


Figure 29. New Linux setup



The command-line setups, in .tar.gz format, are still available for the SDK product. In addition, a new “noarch” installer is available for a platform-independent installation, which can be useful when a COBOL application does not call C functions, hence there is no need to have native components for a pure Java application.

Windows setups have been improved, and they are now in .msi format to provide more flexibility during the installation process. The previous format, which was an .exe, is not available anymore.

### C-Tree RTG v3

isCOBOL Evolve 2021 R1 comes with a new C-tree major release.

C-Tree RTG V3, based on the V12 core technology, provides a series of features and improvements described below.

- Embedded Replication Agent

The Replication Agent is no longer a separate tool to be started, configured and maintained separately. The replication technology is now part of the C-Tree server itself. In order to activate the replication, all that is needed are the following steps, to be carried out on the target C-Tree server before starting it:

- a. Edit the ctsrvr.cfg configuration file and enable the plugin by removing the semicolon before PLUGIN ctagent

- b. Edit the ctreplagent1.cfg configuration file to provide source\_server and target\_server, for example:

```
source_server FAIRCOMS@192.168.1.4  
target_server FAIRCOMS@localhost
```

As always, the C-Tree server must be licensed for replication to be enabled, and the involved files must be under transaction logging.

- New replication features

The data replication can now be either synchronous or asynchronous.

The creation (OPEN OUTPUT) of new indexed files is now replicated as well.

- Browser-Based Tools

The SQL Explorer utility, the Monitor utility and the ISAM Explorer utility, previously available as .NET applications and Java applications, are now also available as web applications. The apps can now be used on platforms where a web-browser is provided, including mobile devices.

To use the web utilities, start the Faircom Web Server (fcWebServer), which is installed along with c-tree. This web server is currently available only for the Windows 64-bit platform and the Linux 64-bit platform. The web server listens for standard HTTP connections on the port 8080 and for secure HTTP connections (HTTPS) on the port 8090. These ports can be changed by editing the chttpd.json configuration file.

Figure 30, Web dashboard, shows the home page of the Faircom's web server, where you can choose which web utility to run.

Figure 30. Web dashboard

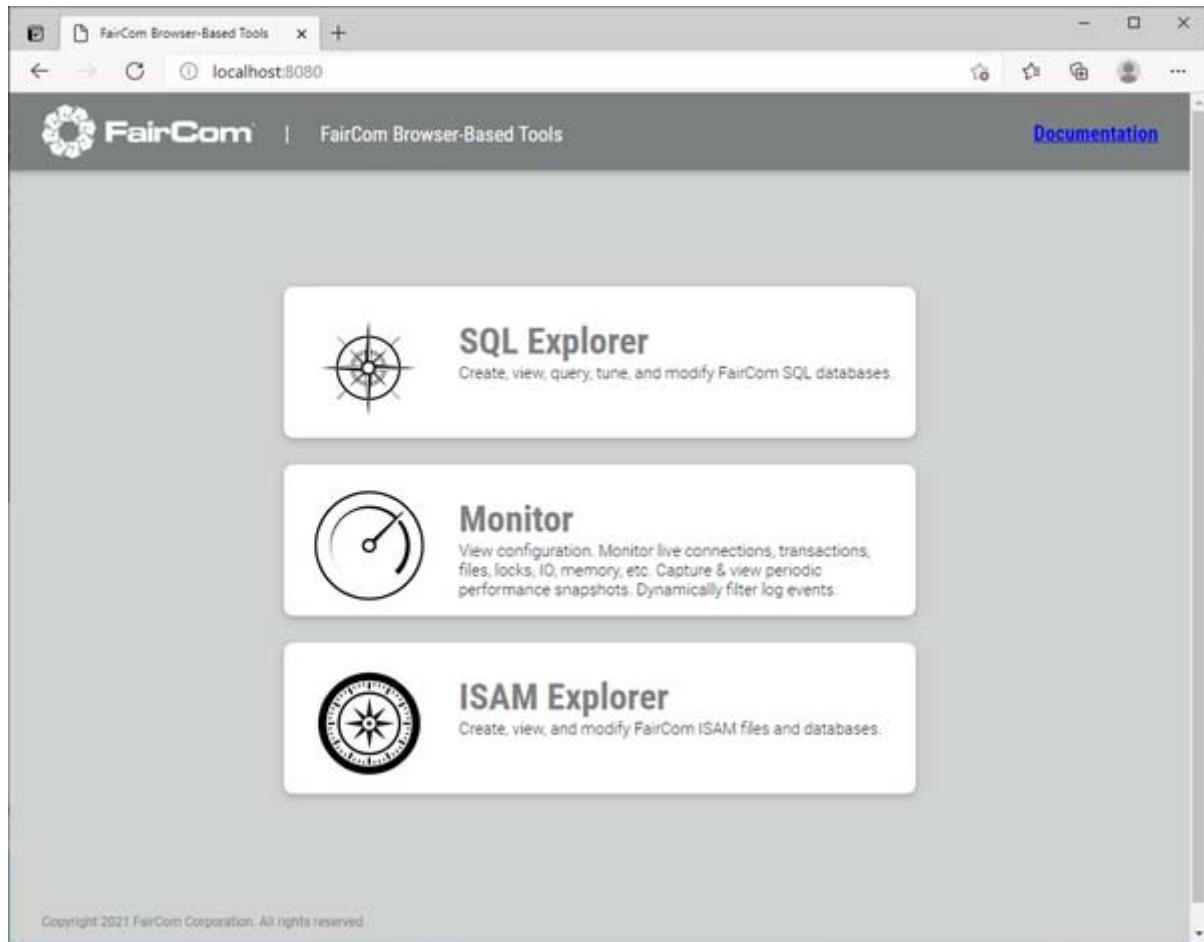


Figure 31, Web SQL Explorer, shows the layout of the browser-based SQL Explorer. This utility allows you to create, view, query, tune, and modify FairCom SQL databases.

Figure 31. Web SQL Explorer

The screenshot shows the FairCom SQL Explorer interface. The left sidebar displays a tree view of the database structure under 'Server' (localhost:6597@localhost). Under the 'admin' database, there are several tables: 'african\_people', 'american\_people', 'asian\_people', 'australian\_people', and 'european\_people'. A table named 'file1' is also listed. The right panel contains a table titled 'Show Number of Rows for each Table' with one row of data. Below this table, there is a detailed view of the 'file1' table's columns, showing five columns: f\_cod, f\_firstname, f\_seename, f\_address, and f\_birthday. The bottom status bar indicates 'Ready' and 'Connected to Database : 6597@localhost:ctreeSQL'.

id (int)	col (varchar)	coltype (varchar)	width (int)	scale (int)	dflt_value (varchar)	nullflag (varchar)	identity (varchar)	seed (int)	incr (int)	charset (varchar)	collation (varchar)
0	f_cod	numeric	3	0	(null)	Y	N				
1	f_firstname	character	20	0	(null)	Y	N				
2	f_seename	character	20	0	(null)	Y	N				
3	f_address	character	20	0	(null)	Y	N				
4	f_birthday	date	4	0	(null)	Y	N				

Figure 32, Web Monitor, shows the layout of the browser-based Monitor. This utility allows you to monitor live connections, transactions, files, locks, IO, memory, etc. It also allows you to view configurations, capture and view periodic performance snapshots and dynamically filter log events.

Figure 32. Web Monitor

The screenshot shows a browser window titled "Monitor" displaying the FairCom Monitor application at "localhost:8080/AceMonitor/index.html". The interface includes a top navigation bar with tabs like "Gauges", "Charts", "Active Connections", "Files / Locks", "Schemas", "ID Performances", "Function Timing", "System Configuration", "Memory Allocations", and "System Events Log". Below this is a section titled "Users" with a table showing two active connections:

#	Task...	User Name	Client IP Addre...	Node ID Info	Last Function	Acti...	Last Request Ti...	Last TRANBEG T...	Logon Time	Fi...	Memory	Conn Info
1	22	ADMIN	127.0.0.1	Web Server - ACEMonitor	ctSNAPSHOT	no	Apr 20, 2021 15...	n.a.	Apr 20, 2021 15...	0	59,128	FSHARE01
2	24	ADMIN (Curre...	127.0.0.1	Web Server - ACEMonitor	USERINFOX	yes	Apr 20, 2021 15...	n.a.	Apr 20, 2021 15...	0	59,168	FSHARE01

Below the table, a note says "Click the + sign or double click a row to view additional info." A detailed "User Snapshot for Task # : 22" table follows:

#	Member	Category	Description	Value	Value / Sec.
1	client_ver	Internal	client version of structure	2	
2	server_ver	Internal	server version of structure	2	
3	fxlen	Internal	length of fixed portion of snapshot	576	
4	varlen	Internal	length of variable region (if any)	0	
5	contents	Internal	bit map of var len contents	0	
6	unused	Internal	available for use	0	
7	snapshottm	Internal	snapshot time stamp: seconds since 70 (H d,Y H:m:s)	April 20, 2021 15:37:17	
8	strtsum	Internal	user trftime sum^ (d - H : m : s)	0 - 00 : 00 : 00	
9	strmax	Internal	user trftime max^ (d - H : m : s)	0 - 00 : 00 : 00	

At the bottom, status bars show "Ready" and "Connected to Server: FAIRCOMS@localhost Last Update Time : 4/20/2021, 3:37:17 PM".

Figure 33, Web ISAM Explorer, shows the layout of the browser-based ISAM Explorer. This utility allows you to create, view, and modify c-tree ISAM files and databases.

Figure 33. Web ISAM Explorer

The screenshot shows a web browser window titled 'localhost:5597/ctreeSQL - admin' with the URL 'localhost:8080/IsamExplorer/index.html'. The main content area is titled 'c-treeACE ISAM Explorer' and displays a database structure under 'Server'. The tree view shows 'FAIRCOMS@localhost' expanded to show 'ctreeSQL' which contains 'file1'. The 'Database' tab is selected, showing 'Table Fields', 'Table Properties', 'Table Indexes', and 'Table Records'. The 'Table Records' section is active, displaying a table titled 'Table : file1 - { Database : ctreeSQL }'. The table has columns: F\_COD (CT\_NUMBER), F\_FIRSTNAME (CT\_PSTRING), F\_SECCNAME (CT\_PSTRING), F\_ADDRESS (CT\_PSTRING), and F\_BIRTHDAY (CT\_DATE). The table shows 52 rows of data, with the first few rows being: 1 Smith Albert 49 West 44th St. 10/11/1971; 2 Johnson Alan 2248 Palm Spring Rd. 01/14/1968; 3 AA BB CC 01/02/1954; 4 AA BB CC 01/02/1954; 5 AA BB CC 01/02/1954; 6 AA BB CC 01/02/1954; 7 AA BB CC 01/02/1954; 8 AA BB CC 01/02/1954; 9 AA BB CC 01/02/1954; 10 AA BB CC 01/02/1954; 11 AA BB CC 01/02/1954; 12 AA BB CC 01/02/1954; 13 AA BB CC 01/02/1954; 14 AA BB CC 01/02/1954; 15 AA BB CC 01/02/1954; 16 AA BB CC 01/02/1954; 17 AA BB CC 01/02/1954; 18 AA BB CC 01/02/1954; 19 AA BB CC 01/02/1954; 20 AA BB CC 01/02/1954; 21 AA BB CC 01/02/1954; 22 AA BB CC 01/02/1954; 23 AA BB CC 01/02/1954; 24 AA BB CC 01/02/1954; 25 AA BB CC 01/02/1954; 26 AA BB CC 01/02/1954.

## IsCOBOL Utilities

The ISMIGRATE utility now shows the tool's progress during the migration of each file. This progress bar is available in both GUI mode and command-line mode. When run in a GUI environment it also shows the estimated remaining time necessary to complete the migration of the file.

As shown in Figure 34, ISMIGRATE GUI progress, the file "customers" is being processed, 40.5 % of the records are migrated and the estimated time remaining is 14 seconds.

Figure 34. ISMIGRATE GUI progress

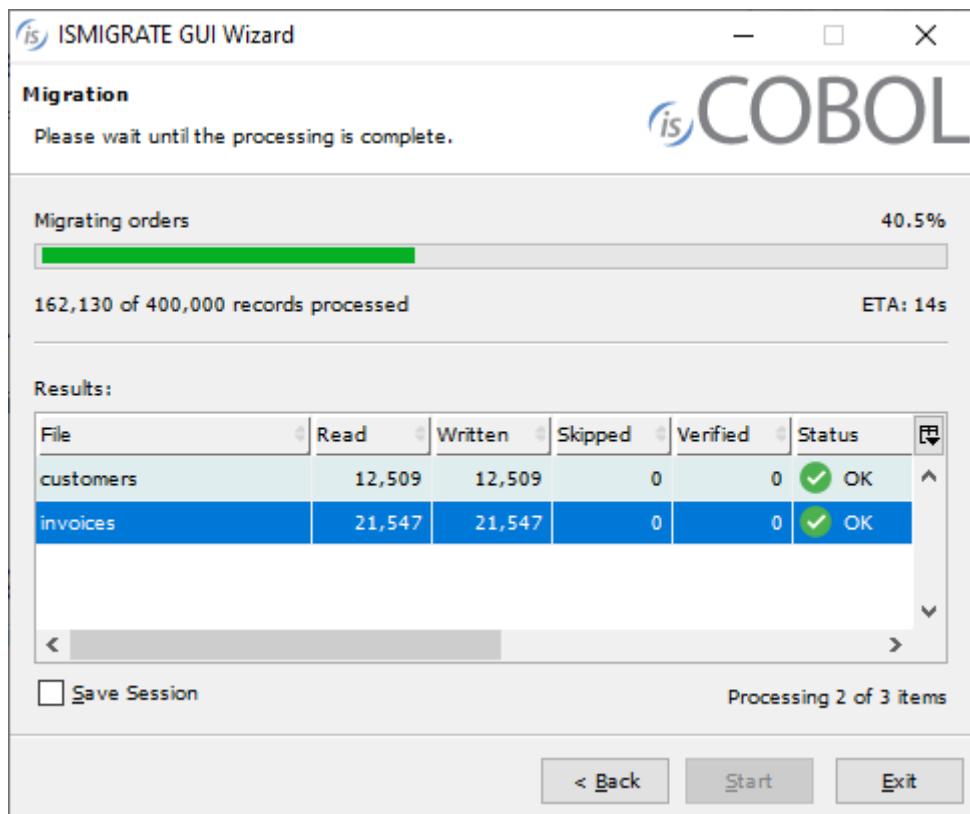


Figure 35, ISMIGRATE command-line progress, shows the same migration running in command-line mode

Figure 35. ISMIGRATE command-line progress

```
C:\WINDOWS\system32\cmd.exe - iscrun -c migrate.properties -utility ISMIGRATE input\*.dat output
C:\data-migration>iscrun -c migrate.properties -utility ISMIGRATE input\*.dat output
(i) Migrating customers.dat [customers.dat]
...10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
(i) Migration OK
(i) 12509 read, 12509 written, 0 skipped, 0 verified

(i) Migrating invoices.dat [invoices.dat]
...10%....20%....30%....40%....50%....60%....70%....80%....90%....100%
(i) Migration OK
(i) 21547 read, 21547 written, 0 skipped, 0 verified

(i) Migrating orders.dat [orders.dat]
...10%....20%....30%....40%....50%....60%...
```

The ISMIGRATE tool also provides the ability to discard records during migration using the configuration option:

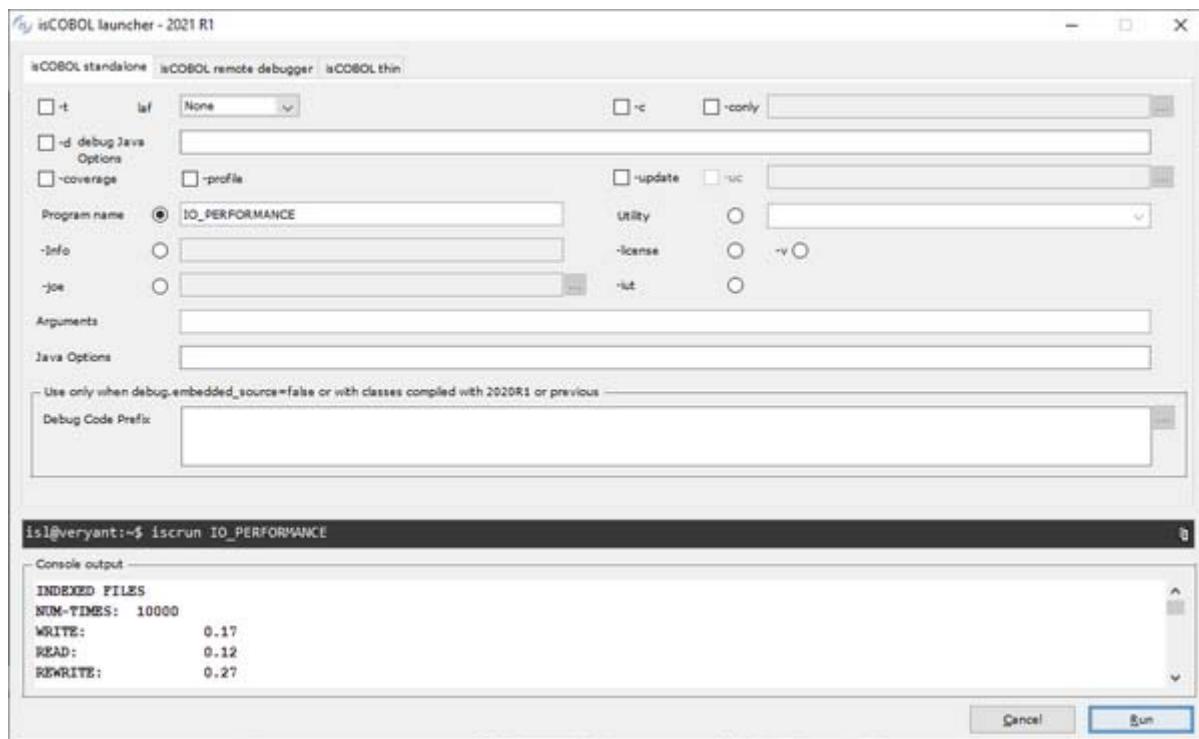
```
iscobol.ismigrate_hook
```

to call a hook program used to filter records. The record being processed is passed in the linkage section of the hook program, and if the hook program exits with a negative return code, for example GOBACK -1, then the record is skipped by ISMIGRATE.

The ISL utility now supports a Console output view, to show the output of iscrun or iscclient when running batch programs. When ISL is running, there is no need to switch to the terminal to see the console output, everything is monitored by the utility.

As shown in Figure 36, ISL console, the output of the batch program IO\_PERFORMANCE is shown in the new Console output section at the bottom of the screen.

Figure 36. ISL console



# isCOBOL 2020 Release 2 Overview

## Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2020 R2.

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

A new isCOBOL Profiler makes it easier to identify application bottlenecks.

The 2020 R2 release has new GUI controls such as the new “hamburger menu” and glyph fonts rendering, and improves many existing ones, such as grids and message boxes, to further help developers modernize their applications.

The Debugger has been reengineered, and now source code can be embedded in the compiled class, making debugging sessions possible without having the source. The new debugger also adds a new Console View that separates messages coming from sysout and syserr from other debugger output.

Details on these enhancements and updates are included below.

## New isCOBOL Profiler

Starting with the 2020R2 Release, a new Profiler has been implemented to allow performance analysis on applications, making it easy to identify which program or paragraphs take longer to execute. Developers can view the gathered information in a clean HTML output, and take actions to perform code optimizations accordingly to maximize gains.

This feature is now available in the isCOBOL Evolve suite, and can be accessed from the command line or from the Eclipse-based isCOBOL IDE.

The Profiler can be enabled in different ways depending on deployment architecture. It can be activated using a runtime option, as shown in the command line below:

```
iscrun -profile IO_PERFORMANCE
```

With this command, the isCOBOL Profiler creates a folder called “hprofHtmlReport” containing an HTML report of the profile analysis. Figure 1, isCOBOL Profile report, shows the report with a list of all executed programs and the percentage of the total run time, as well as individual paragraph profile results.

The report shows both the percentage of total time and the number of seconds spent in each programs’ paragraphs. It also shows how many times a paragraph has been executed in the “Count” column.

Another way to start the profiling process, compatible with the previous profiler feature, is using a java option:

```
-javaagent:/Veryant/isCOBOL2020R2/lib/isprofiler.jar
```

This method of activating the profiler is most useful when running in an Application Server architecture, or in scenarios such as Java programs calling isCOBOL or running isCOBOL EIS applications in a J2EE container like Apache Tomcat, when the main class is not the COBOL program.

In isCOBOL 2020 R2, the Profiler can also be activated or stopped dynamically at runtime, by performing a CALL "C\$PROFILE" instruction. An example of this usage will be shown in a later section of this document.

Figure 1. isCOBOL Profile report

The screenshot shows a Microsoft Edge browser window titled 'isCOBOL Profile Report'. The URL in the address bar is 'file:///C:/isCOBOL2020R2/sample/io-performance/hprofHtmlReport/index.htm'. The page content is titled 'isCOBOL Profile Report' and includes the following information:

- Executed: July 1, 2020 12:55:13 PM
- Elapsed: 3.144s; Evaluated: 2.919s; Overhead1: 41; Overhead2: 43
- Warning: some programs are compiled in debug mode, this affects performances.
- Search: Filter...
- [View Program table](#)

Program	Paragraph	Self %	Seconds	Count
IO_INDEXED	DELETE_FILE1_TEST	28.38%	0.828	1
IO_INDEXED	UPDATE_FILE1_TEST	15.94%	0.465	1
IO_INDEXED	LOAD_FILE1_TEST	11.33%	0.331	1
IO_LINESEQUENTIAL	LOAD_FILE1_TEST	6.81%	0.199	1
IO_INDEXED	READ_FILE1_TEST	6.59%	0.192	1
IO_RELATIVE	DELETE_FILE1_TEST	4.65%	0.136	1
IO_RELATIVE	UPDATE_FILE1_TEST	4.37%	0.128	1
IO_RELATIVE	LOAD_FILE1_TEST	3.62%	0.106	1
IO_SEQUENTIAL	LOAD_FILE1_TEST	3.61%	0.105	1
IO_LINESEQUENTIAL	READ_FILE1_TEST	3.05%	0.089	1
IO_PERFORMANCE	MAIN_LOGIC	2.88%	0.084	1
IO_RELATIVE	READ_FILE1_TEST	1.67%	0.049	1
IO_RELATIVE	START_TIMER	1.58%	0.046	4
IO_SEQUENTIAL	READ_FILE1_TEST	1.47%	0.043	1
IO_INDEXED	START_TIMER	1.27%	0.037	4
IO_INDEXED	MAIN_LOGIC	1.04%	0.03	1
IO_SEQUENTIAL	START_TIMER	0.89%	0.026	2
IO_LINESEQUENTIAL	START_TIMER	0.54%	0.016	2
IO_RELATIVE	MAIN_LOGIC	0.13%	0.004	1
IO_LINESEQUENTIAL	MAIN_LOGIC	0.08%	0.002	1
IO_SEQUENTIAL	MAIN_LOGIC	0.07%	0.002	1
IO_INDEXED	STOP_TIMER	0.02%	0.001	4
IO_RELATIVE	STOP_TIMER	0.01%	0	4
IO_SEQUENTIAL	STOP_TIMER	0.00%	0	2
IO_LINESEQUENTIAL	STOP_TIMER	0.00%	0	2
Totals:		100.00%	2.919	

[View Program table](#)

Figure 2. isCOBOL Profile program table

The screenshot shows a web browser window titled "isCOBOL Profile Report". The URL in the address bar is "file:///C:/isCOBOL2020R2/sample/io-performance/hprofHtmlReport/indexpg". The page content includes the title "isCOBOL Profile Report", execution date "Executed: July 1, 2020 12:55:13 PM", and performance metrics "Elapsed: 3.144s; Evaluated: 2.919s; Overhead1: 41; Overhead2: 43". A warning message states "Warning: some programs are compiled in debug mode, this affects performances". A search bar is present with the placeholder "Search: Filter...". Below the search bar is a link "View Paragraph table". A table titled "Program" lists the following data:

Program	Self %	Seconds	Count
IO_INDEXED	64.58%	1.885	1
IO_RELATIVE	16.02%	0.468	1
IO_LINESEQUENTIAL	10.48%	0.306	1
IO_SEQUENTIAL	6.04%	0.176	1
IO_PERFORMANCE	2.88%	0.084	1
Totals:	100.00%	2.919	

At the bottom of the table is another link "View Paragraph table".

Clicking on a source file name, for example IO\_INDEXED, the Paragraph table view is loaded, with the filter automatically set to the selected program, thus showing all its paragraphs performance details.

In Figure 3, isCOBOL Profile filter, shows how to filter the output by program name.

Figure 3. isCOBOL Profile filter

The screenshot shows a web browser window titled "isCOBOL Profile Report". The URL in the address bar is "file:///C:/isCOBOL2020R2/sample/io-performance/hprofHtmlReport/index.htm". The page content includes the title "isCOBOL Profile Report", execution date "Executed: July 1, 2020 12:55:13 PM", and performance metrics "Elapsed: 3.144s; Evaluated: 2.919s; Overhead1: 41; Overhead2: 43". A warning message states "Warning: some programs are compiled in debug mode, this affects performances". A search bar is present with the placeholder "Search: IO\_INDEXED". Below the search bar is a link "View Program table". A table titled "Program" lists the following data:

Program	Paragraph	Self %	Seconds	Count
IO_INDEXED	DELETE_FILE1_TEST	28.38%	0.828	1
IO_INDEXED	UPDATE_FILE1_TEST	15.94%	0.465	1
IO_INDEXED	LOAD_FILE1_TEST	11.33%	0.331	1
IO_INDEXED	READ_FILE1_TEST	6.59%	0.192	1
IO_INDEXED	START_TIMER	1.27%	0.037	4
IO_INDEXED	MAIN_LOGIC	1.04%	0.03	1
IO_INDEXED	STOP_TIMER	0.02%	0.001	4
Totals:		64.57%	1.884	

At the bottom of the table is another link "View Program table".

The report can be filtered by typing in the Search field, to select specific programs or paragraphs. The filter is applied to all columns in the table. The report shows a warning if programs are profiled when compiled in debug mode, or if logging is enabled, as that will impact performance. Should this occur, developers should recompile the programs without the debug option, and run with the `iscobol.tracelevel` configuration option disabled in order to profile with maximum performance.

Profiler reports can be customized using new configuration options:

- `iscobol.profiler.html=path` sets the folder in which isCOBOL Profiler will create the report. If not set, the default value is `./hprofHtmlReport`
- `iscobol.profiler.xml=path` sets the file path of xml report. If not set, the xml report is not generated. This is needed when running the isCOBOL Profiler inside the IDE to take advantage of its specific View.
- `iscobol.profiler.excludes=path` sets the list of isCOBOL classes that will be excluded from the profiling analysis. It is a comma-separated list of regular expressions defining the programs, for example `IO_INDEXED,IO_RELATIV[A-Z_]`
- `iscobol.profiler.includes=path` sets the list of isCOBOL classes that will be included in the profiling analysis. It uses the same syntax explained above.

The new library routine C\$PROFILE is handy when trying to profile a program that contains ACCEPT statements, since performance analysis is affected by the time the user takes to complete the ACCEPT statement. Developers can easily suspend the profiling process when such events occur, and resume it when it's done. The routine also allows you to dynamically set the report format and the folder in which to create the report.

The code snippet below shows how to perform all the steps explained above from code:

```
call "C$PROFILER" USING cprof-disable
accept mask-parameters
call "C$PROFILER" using cprof-set "html" "html-profiler-output"
call "C$PROFILER" USING cprof-enable
perform elaboration.
call "C$PROFILER" USING cprof-disable
display message "processing completed"
call "C$PROFILER" USING cprof-flush
```

The Code Coverage options have analogous settings to control its behavior at runtime using the C\$COVERAGE library routine with new op-codes, as shown in the snippet below:

```
call "C$COVERAGE" using ccov-set "html" "html-coverage-output"
...
call "C$COVERAGE" USING ccov-flush
```

The Code Coverage feature, which was previously available when running stand-alone programs using iscrun, is now fully supported in other environments, such as the Application Server, using the new property options for `-javaagent`, as shown below:

```
-javaagent:/Veryant/isCOBOL2020R2/lib/isprofiler.jar=coverage;html=html-
coverage;profiler;html=html-profiler
```

When run with this option, the Application Server, upon termination, will create the Code Coverage and the Profiler reports in two different folders, unless the C\$PROFILER or C\$COVERAGE routines are called by running programs to flush the reports.

The `-javaagent` option enables the customization of additional parameters of both Code Coverage and Profiler features:

- `includes=` comma separated list of regular expressions to specify the classes to be included in the report

- excludes= comma separated list of regular expressions to specify the classes to be excluded from the report
- xml= name of the xml output file

Additional options are available for the Coverage feature:

- sourcefiles= the location of the source files
- classfiles= the location of the class files
- append= the xml file to be appended to the output. This can be specified multiple times
- sessionname= the name of the coverage session

All these features enhance Veryant products and complete the isCOBOL Code Coverage and isCOBOL Unit Test features introduced in the 2020 R1 release, and are especially appealing to enterprise users.

## isCOBOL IDE enhancements

The Profiler feature can now be executed from the isCOBOL 2020 R2 IDE, and a specific View is provided to show and analyze the results of the profiling process.

Figure 6, isCOBOL Profiler in isCOBOL IDE shows how to enable profiling on specific programs. By default, all programs are included in Profiler, and the results gathered by the analysis are available in the Profiler View, as shown in Figure 7, isCOBOL Profiler View.

Figure 6. isCOBOL Profiler in isCOBOL IDE

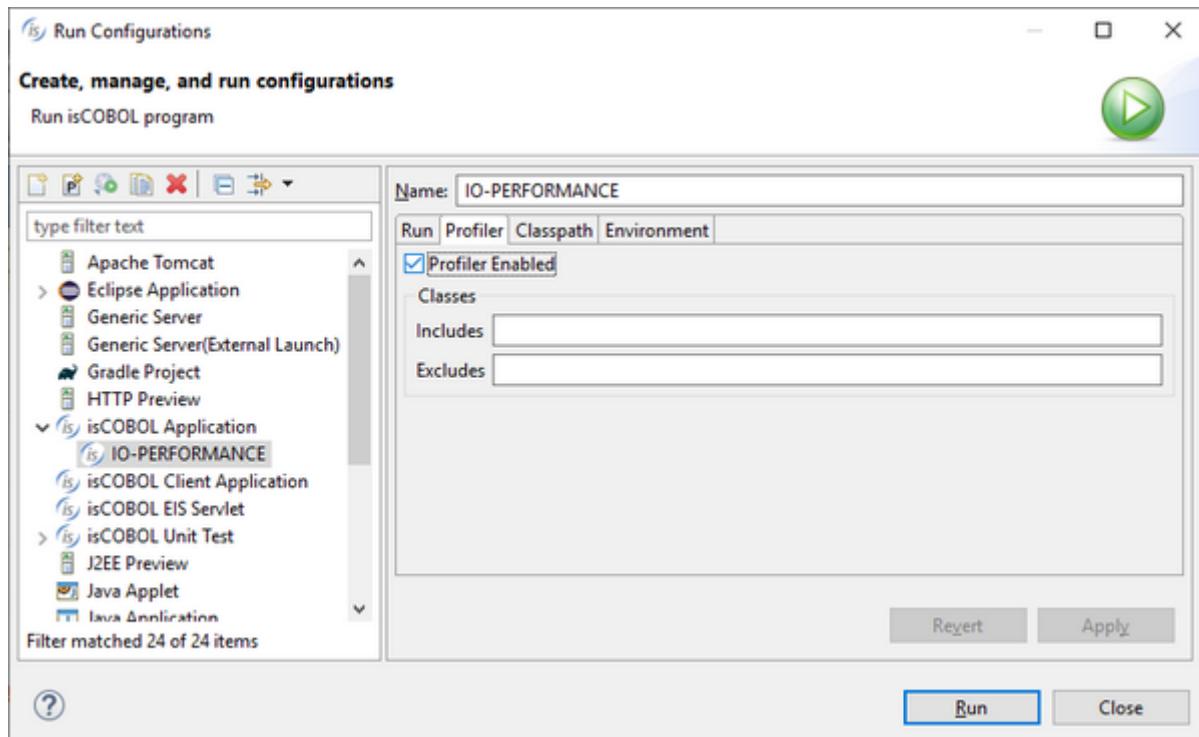


Figure 7. isCOBOL Profiler View

Program	Paragraph	Self %	Total time	Count
IO_INDEXED	DELETE_FILE1_TEST	961.890 ms (27.60%)	1	
IO_INDEXED	UPDATE_FILE1_TEST	596.798 ms (17.12%)	1	
IO_INDEXED	LOAD_FILE1_TEST	362.535 ms (10.40%)	1	
IO_INDEXED	READ_FILE1_TEST	274.935 ms (7.89%)	1	
IO_LINESEQUENTIAL	LOAD_FILE1_TEST	217.169 ms (6.23%)	1	
IO_RELATIVE	UPDATE_FILE1_TEST	206.157 ms (5.91%)	1	

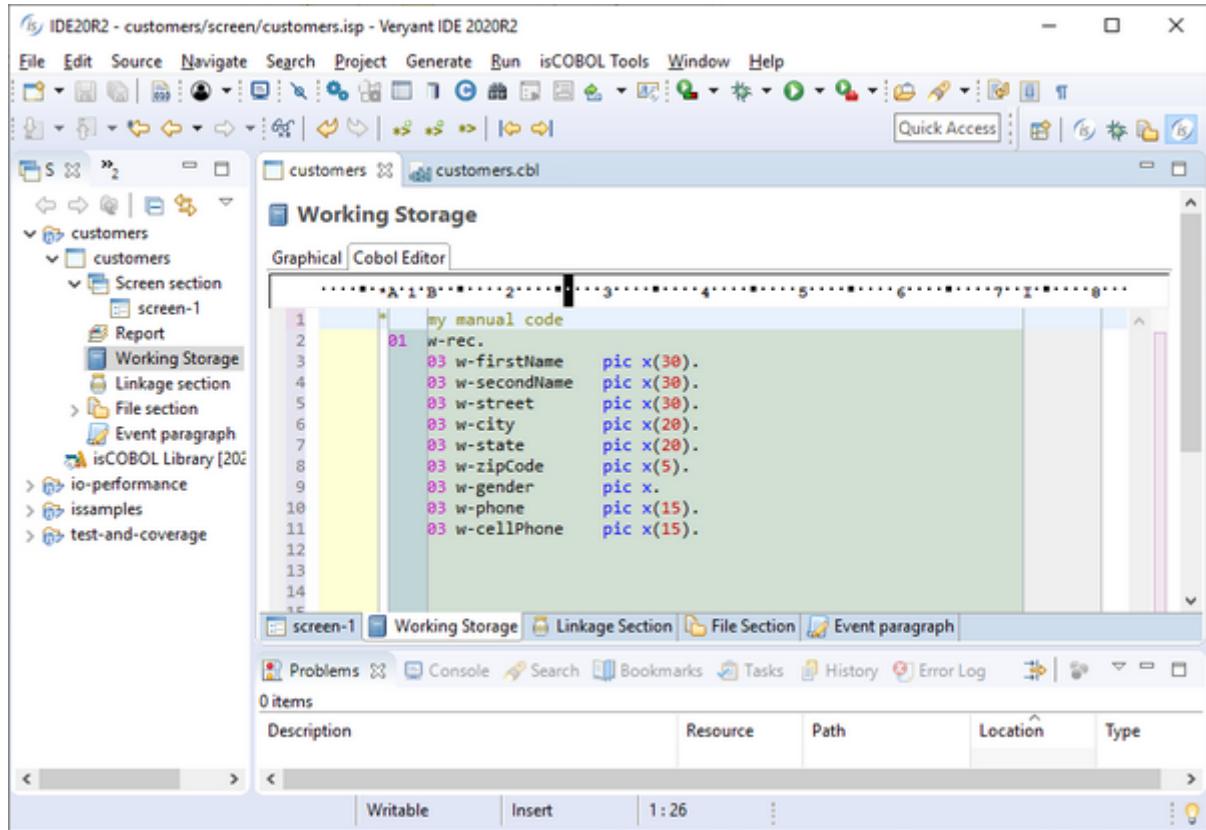
Double clicking an item in the Profiler View opens the selected program in the Editor, with the cursor located at the specific paragraph, helping developers speed up the process of identifying bottlenecks to optimize the code for better performance.

The isCOBOL IDE's Screen Painter now supports manual editing in Working Storage, Linkage Section and FD sections, to complement the Graphical editor previously available.

Changes made manually are available in the Screen Painter immediately. This feature will be appreciated by COBOL developers who would prefer to code by hand instead of through the Graphical editor.

Figure 8, isCOBOL IDE Cobol Editor, shows the Cobol Editor section of the Working Storage painter, where a group level variable is manually coded. The Painter will append the manual code fragment in the generated source code after the variables defined in the Graphical editor.

Figure 8. isCOBOL IDE Cobol Editor



## GUI enhancements

GUI controls have been improved in the 2020 R2 release. A new menu style is now available, the Hamburger menu, that arranges its items in a tree-view style.

The W\$BITMAP library routine can now render glyph font characters as images that can be used as icons in any control that can handle bitmaps.

### Hamburger menu

The isCOBOL compiler now supports a new menu style that arranges items in a hierarchical tree-view which appears after pressing an “hamburger” icon, and can replace the classic menu bar. This new feature can be used to give applications a more modern look, or to replace a standard menu bar in a responsive application to adapt to a resized surface area.

A new op-code, wmenu-new-hamburger, has been added to the W\$MENU library routine to enable the creation of hamburger menus.

A code snippet to create a hamburger menu is shown below:

```
call "W$MENU" using wmenu-set-attribute
      "default-font" h-menu-font
call "W$MENU" using wmenu-set-attribute
      "width" 160
call "W$MENU" using wmenu-new-hamburger giving menu-handle
```

The new wmenu-set-attribute op-code allows full customization of the graphical appearance of the menu, such as fonts, colors and icons. The full list of supported attributes is: check-icon, default-background-color, default-font, default-text-color, disabled-background-color, disabled-font, disabled-text-color, dropdown-icon, dropdown-open-icon, hamburger-icon, hamburger-open-icon, hover-background-color, hover-font, hover-text-color, position, tool-bar-covering, style, width.

Existing W\$MENU op-codes can be used to manage the hamburger menu the same way as regular menus.

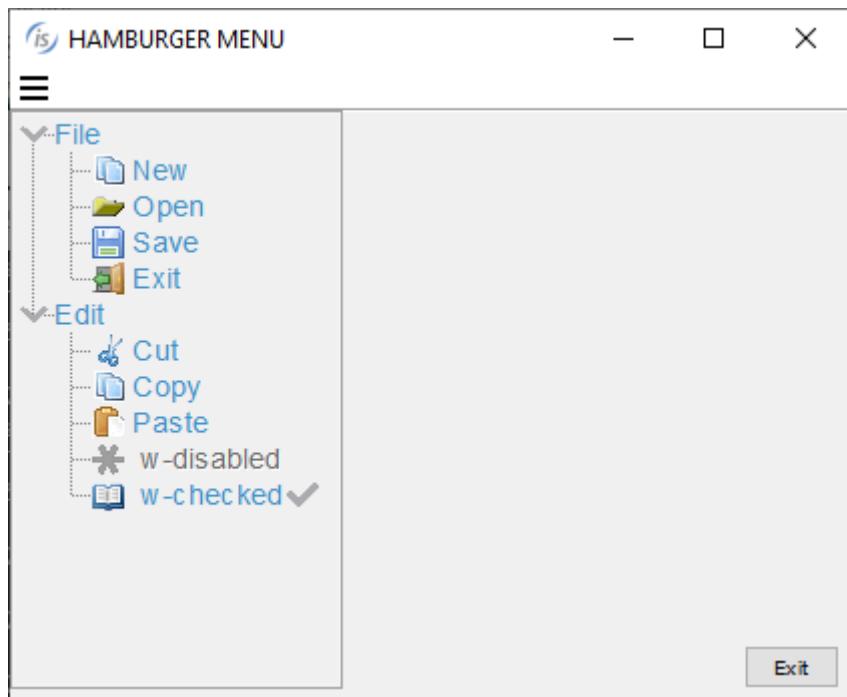
A new attribute called "menu-bar-flavor" has been implemented to easily change the default menu style of the application. Executing the code

```
call "W$MENU" using wmenu-set-attribute "menu-bar-flavor" "hamburger"
```

sets the hamburger menu as default menu type, and the entire application will use hamburger menus instead of the classic menu-bar.

Figure 9, Hamburger menu, shows a program running with a hamburger menu opened after pressing the hamburger button.

Figure 9. Hamburger menu



## Glyph fonts

Glyph fonts are vector fonts that contain icons instead of characters, and have been widely used in web applications. Being vector-based make them especially suitable to scale without visual quality loss when used on high-DPI displays.

In the isCOBOL 2020 R2 release they are available for COBOL programs, using the new "wbitmap-load-symbol-font" op-code of the W\$BITMAP library routine. Using this op-code, text and symbols can be converted to bitmaps, and can then be used on any control that supports bitmap-handle and bitmap-number properties, such as push-button, entry-field, grid, tree-view.

The following code snippet loads a font using the W\$FONT routine, then it loads the glyph symbols of 3 characters identified by their Unicode numbers. Using the new op-code, the W\$BITMAP routine can be called passing the font handle, the list of characters, the font-size and color to be used. The routine will generate a bitmap strip that can be used in controls that can handle bitmaps strips,

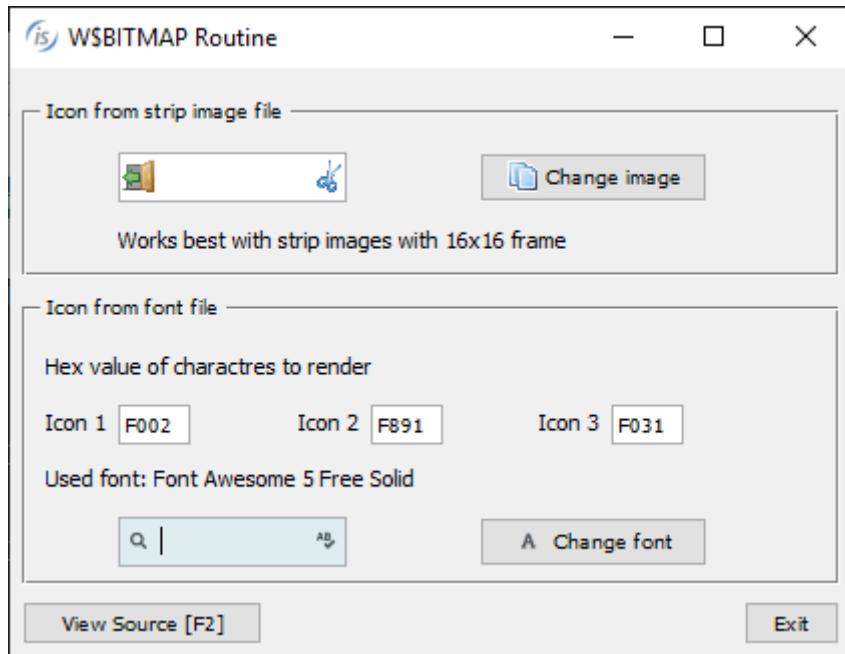
```

working-storage section.
 77 h-font                      handle of font.
 77 h-font-icon                 pic s9(9) comp-4.
 77 icon-size                   pic 9(2).
 77 icon-color                  pic s9(9).
 77 icon-characters            pic n any length.
...
screen section.
01 Mask.
...
 03 ef-font entry-field
    bitmap-handle h-font-icon bitmap-width 16
    bitmap-number 1  bitmap-trailing-number 2
...
 03 pb-font push-button title "Change font"
    bitmap-handle h-font-icon bitmap-width 16
    title-position 2 bitmap-number 3
...
procedure division.
...
  move "Font Awesome 5 Free Solid" to font-name
  call "w$createfont" using "Font Awesome 5 Free-Solid-900.otf" font-name
  initialize wfont-data
  set wfdevice-console to true
  move font-name      to wfont-name
  move 10           to wfont-size
  call "W$FONT" using wfont-get-font h-font wfont-data
  move nx"f002f891f031" to icon-characters
  call "w$bitmap" using wbitmap-load-symbol-font h-font
                    icon-characters icon-size icon-color
                    giving h-font-icon
  display Mask
...

```

The final result is shown in Figure 10, Controls with glyph symbols, where the zoom icon is bitmap-number 1, and is loaded from the “Awesome 5 Free-Solid-900.otf” font, and has Unicode character nx“F002”.

Figure 10. Controls with glyph symbols



This new feature makes it easy to enhance isCOBOL GUI applications to use glyph fonts as icons instead of loading bitmap images from disk.

### Other GUI enhancements

The Grid control has been enhanced by adding a new filter-types property to better manage filter types for each column.

It's now possible to specify at a column level if the filter feature is available and the type of filter, as shown in the following code:

```
filter-types (0, 2, 0, 2, 1, 2, 2, 1)
```

A value of 0 will disable filtering for the column, 1 will remove the filter after a click on the red funnel icon, and a value of 2 will reopen the previously defined filter when clicking the yellow funnel icon.

The search panel has been enhanced by adding the ability to search with case insensitive rules. All buttons in the panel have now an icon.

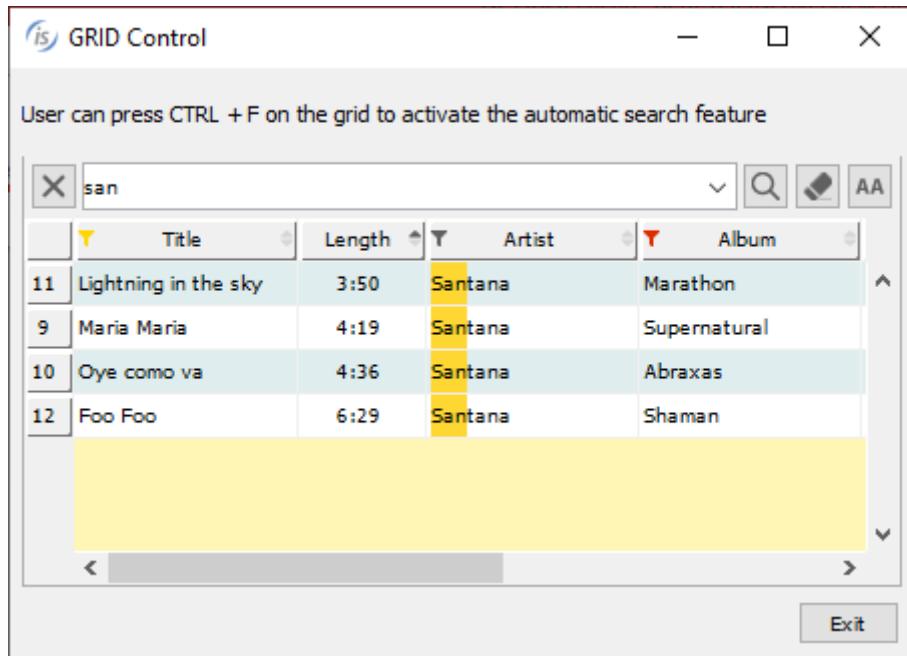
The search panel can now be opened programmatically using the new action-search value, as shown in the code below:

```
modify my-grid action action-search
```

The sort feature activated with sortable-columns or sort-types properties now have new icons. All grid icons can be customized by adding a JAR file that contains the image files to the CLASSPATH.

All these new grid features are shown in Figure 11, Grid filter, sort and search features.

Figure 11. Grid filter, sort and search features



The tree-view control now supports selecting multiple elements of the same level. This can be activated with the selection-mode property as shown in the code snippet:

```
selection-mode tvsm-multiple-interval-selection
```

With this feature enabled, multiple items can be selected using the usual Control and Alt key combinations.

Selected items can be inquired by the program using the items-selected property that returns the list of item handles:

```
inquire tv1 items-selected tv-items-sel
```

Figure 12, Tree-View selection-mode, shows the new tree-view feature in action.

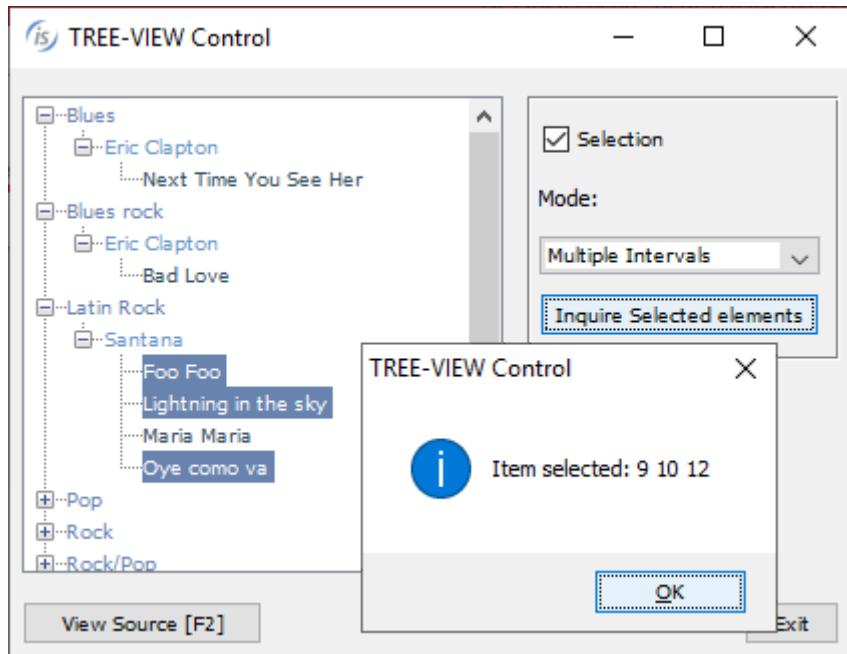


Figure 12 also shows the new message box, which has been revamped to provide a more modern look and feel. Additionally, the message box can now be centered in the monitor instead of the parent window using the CENTERED keyword, as shown in the following code snippet.

```
display message "Are you sure to continue?"  
    title "Confirmation"  
    type mb-yes-no  
    icon mb-warning-icon  
    default mb-no  
    timeout 800  
    centered  
    giving mb-return
```

A new configuration property can be used to automatically apply the new CENTERED option at the application level:

```
iscobol.gui.messagebox.centered=true
```

It is also possible to provide a custom message box implementation using a new configuration property:

```
iscobol.gui.messagebox.custom_prog=PROGRAM_NAME
```

PROGRAM\_NAME is a compiled COBOL program that is automatically called by the isCOBOL framework every time a display message statement is executed. The program receives the details of the message box as linkage section parameters as defined below, and can provide a customized implementation:

```
linkage section.  
77 msgbox-text          pic x any length.  
77 msgbox-title         pic x any length.  
77 msgbox-type          pic 9.  
77 msgbox-icon           pic 9.  
77 msgbox-default        pic 9.  
77 msgbox-timeout        pic 9(5).  
77 msgbox-centered       pic 9.  
procedure division using msgbox-text  
                      msgbox-title  
                      msgbox-type  
                      msgbox-icon  
                      msgbox-btn-default  
                      msgbox-timeout  
                      msgbox-centered  
.
```

The web-browser control can now be used with the JxBrowser Java component, a powerful Chromium-based integrated browser for Java applications that processes and displays HTML5, CSS3, JavaScript, Flash etc. Customers that used JxBrowser in Java applications can now use it in isCOBOL applications by setting the following configuration:

```
iscobol.gui.webbrowser.class=com.iscobol.browser.jx.JXWebBrowser?licenseKey=...
```

## isCOBOL Compiler

The isCOBOL 2020R2 compiler is faster in massive compilations, a new EasyLinkage feature has been implemented to easily convert C calls into pure Java libraries and additional syntax is supported to increase compatibility with other COBOL dialects.

### Performance on massive compilations

The compiler has been reworked by removing the need for the tools.jar file to be in CLASSPATH, and the compilation of .java files to .class objects is now faster, especially when performing large command line compilations, such as when using the command:

```
iscc -options... source/*.cbl
```

Figure 13, Comparison of compiler performance, shows a performance comparison between isCOBOL 2019R1, 2019R2, 2020R1 and the current 2020R2. The tests were run using Windows 10 Pro 64-bit on an Intel Core i-7 Processor-8550U, 1.80 GHz with 16 GB of RAM, using Oracle JDK 1.8.0\_251. All times are in seconds. The test was compiling a real application module consisting of 550 programs with more than 10,000 copy files, for a total of over 1 million lines of COBOL code. Different compiler options have been used depending on the type of compilation.

Figure 13. Comparison of compiler performance

type of compilation	2019R1	2019R2	2020R1	2020R2
release mode -ostrip	641	536	522	430
debug mode -d	784	706	675	585
debug mode -dx	1349	1083	1082	834
debug mode -dx and .list with -If	1570	1190	1181	878

isCOBOL is constantly being improved for performance and customers using older versions now have another good reason to upgrade their release to take advantage of faster compiling times.

## EasyLinkage for Java migration

When migrating COBOL code that relies on C code where COBOL code makes native function calls using the CALL statement, it is usually advised that you migrate the C code to Java to have more readable code and avoid depending on C. To help in the process, the new isCOBOL 2020 R2 compiler introduces a new feature that generates Java stub sources to handle COBOL parameters received in CALL statements.

Assume, for example, the following COBOL code, which uses a C library with 3 different functions that are being ported to isCOBOL and the C library is being migrated to Java:

```

WORKING-STORAGE SECTION.
 77 w-path      pic x(256).
 77 path-len    pic 9(3).
 01 w-exec.
   03 w-command  pic x(128).
   03 w-options  pic x(64).
   03 w-param    pic 9(2) comp.
...
PROCEDURE DIVISION.
MAIN.
...
call "MyLibrary"
call "func_initialize"
call "func_get_path" using by value path-len
                     by reference w-path
call "func_exec" using w-exec
...

```

The source code can be compiled using the new configuration setting:

```
iscobol.compiler.easylinkage=2
```

to automatically generate the stub sources called: MYLIBRARY.java, FUNC\_INITIALIZE.java, FUNC\_GET\_PATH.java and FUNC\_EXEC.java. The main problem when integrating COBOL code with another language is understanding how to code the proper memory structures to receive the COBOL parameters: how are group variables handled? How is a comp variable stored?

When compiling with the above property set, the isCOBOL compiler takes care of this complexity by generating the correct parameter definitions in the Java source code. Developers only need to focus on porting the C logic to Java. The EasyLinkage feature even handles C functions with variable numbers of arguments and different sizes.

For example, in the FUNC\_EXEC.java source the COBOL group level w-exec has been declared as:

```
// variable declaration W-EXEC
public byte wExec$0 [];
public com.iscobol.types.PicX wExec;
public com.iscobol.types.PicX wCommand;
public com.iscobol.types.PicX wOptions;
public com.iscobol.types.NumericVar wParam;
...
{
    wExec$0=Factory.getMem(194);
    wExec=Factory.getVarAlphanum(wExec$0,0,194,...,"W-EXEC",...);
    wCommand=Factory.getVarAlphanum(wExec,0,128,...,"W-COMMAND",);
    wOptions=Factory.getVarAlphanum(wExec,128,64,...,"W-OPTIONS",...);
    wParam=Factory.getVarBinary(wExec,192,2,...,"W-PARAM",...);
}
...
/* Write the routine logic here
...
```

Additionally, the compiler generates by default the source files in specific subfolders called:

- “easydb” for database bridge generation (any kind of database option)
- “easylinkage” for EasyLinkage generation (both link and stub)
- “servicebridge” for web service bridge generation (both REST and SOAP)
- “bean” for web service bean generation

By default, these folders are generated inside the sources folder, but can be configured using the following compiler property:

```
iscobol.compiler.generate.root_dir=path
```

The .class files created by compiling the generated COBOL sources (easydb, servicebridge and bean) are by default located in the same folder where the main COBOL program .class is created, in the path specified using the –od compiler option. It is also possible to create the same subfolder names under the main class folder by using: iscobol.compiler.generate.keep\_structure=true, resulting in a clearer organization of folders.

## Compatibility

The compiler now supports the EXHIBIT statement when using the –cv option for IBM COBOL compatibility. Running the code below

```
MOVE "ABC" TO MY-VAR
EXHIBIT NAMED MY-VAR
```

when compiled using the –cv option produces the following output on sysout when run:

```
MY-VAR=ABC
```

This is similar to the DISPLAY UPON SYSOUT statement, but it's now fully supported and doesn't require code to be manually changed when porting to isCOBOL.

The ESQL DECLARE CURSOR statement has been enhanced to support the SENSITIVE and INSENSITIVE clauses. Code such as:

```
exec sql
    declare curs1 sensitive cursor for
        select * from companies
end-exec.
exec sql
    declare curs2 insensitive cursor for
        select * from companies
end-exec.
```

are now supported and can be compiled to simplify the migration from other COBOL ESQL Pre-compilers.

Inensitive allows the cursor to move forward and backward through the data, changes made while the cursor is open are ignored. Sensitive allows the cursor to move forward and backwards through the data, changes made while the cursor is open are immediately available.

This behavior is similar to the runtime configuration `iscobol.jdbc.cursor.type`, and both can be used in the same program, allowing additional flexibility.

## isCOBOL Runtime

Starting from isCOBOL 2020R2, new configuration settings and new library routines are available to customize behavior and improve compatibility.

The new configuration settings are:

- `iscobol.key.<keystroke>=search=<context>` to customize the search action in a specific component. By default, pressing Ctrl+F on grid, print-preview and web-browser makes the Search panel appear. The key combination can be customized: for example it can be set to Shift+F4 by using the following configuration settings:
- `iscobol.key.*f=search=`
- `iscobol.key.^f4=search=grid,print-preview,web-browser`

If you need to disable the Ctrl+F feature for the web-browser only, the following setting can be used:

```
iscobol.key.*f=search=print-preview,grid
```

- `iscobol.call_cancel.hook=className` to specify a hook class for CALL and CANCEL statements. The class can be written in pure Java or in COBOL using the appropriate syntax CLASS-ID to implement the interface “com.iscobol.rts.CallHandler”. The list of required methods that need to be defined for this interface are:
  - `public void afterCall(IscobolCall call, Object[] argv);`
  - `public void afterCancel(IscobolCall call);`
  - `public void afterCancelAll();`
  - `public void beforeCall(IscobolCall call, Object[] argv);`
  - `public boolean beforeCancel(IscobolCall call);`

- o public boolean beforeCancelAll();
- iscobel.jdbc.auto\_connect=true to automatically connect to the database without using an explicit ESQL CONNECT statement in the program. When the first ESQL statement is executed, the connection is automatically established.
- iscobel.jdbc.user=userName to supply a username credential without specifying it in the iscobel.jdbc.url property. This can be used also in conjunction with iscobel.jdbc.auto\_connect=true
- iscobel.jdbc.password=pwd to supply the connection password without specifying it in the iscobel.jdbc.url property. This can be used also in conjunction with iscobel.jdbc.auto\_connect=true

New library routines have been implemented to enhance the compatibility with the MicroFocus® dialect:

- CBL\_ALLOC\_MEM to allocate memory. The allocated memory amount is expressed in bytes, and it returns a pointer used to release allocated memory.
- CBL\_FREE\_MEM to release the previously allocated memory.
- CBL\_GET\_CURRENT\_DIR to retrieve the current folder name.
- CBL\_READ\_DIR to retrieve the current folder absolute path name.

The following is a code snippet of a migrated COBOL program that calls all the new supported library routines:

```

WORKING-STORAGE SECTION.
 77 path-name          pic x(256).
 77 path-name-length   pic x comp-x.
 77 dir-status-code    pic x(2) comp-5.
 77 flags              pic x(4) comp-5.
 77 name-length        pic x(4) comp-5.
 77 mem-pointer        usage pointer.
 77 mem-size           pic x(4) comp-5.
 77 mem-flags          pic x(4) comp-5.
 77 mem-status-code    pic x(2) comp-5.
...
PROCEDURE DIVISION.
...
  call "CBL_READ_DIR" using path-name
                  path-name-length
                  returning dir-status-code
  call "CBL_GET_CURRENT_DIR" using by value      flags
                                by value      name-length
                                by reference path-name
                                returning     dir-status-code
  call "CBL_ALLOC_MEM"  using                 mem-pointer
                                by value mem-size
                                by value mem-flags
                                returning mem-status-code
  call "CBL_FREE_MEM"   using by value mem-pointer
                                returning mem-status-code

```

## isCOBOL Debugger

Starting from isCOBOL 2020R2, the isCOBOL Debugger has been enhanced and can now debug programs without having access to source files. When compiled with the -d or -dx options, the compiler now embeds an encrypted copy of the source code in the compiled .class file. Such a .class file can now be fully debugged anywhere. This is completely transparent for developers, who will have access to the COBOL source during a debugging session as they had previously, and can take advantage of the same debugger commands that need references to source code lines, such:

```
br 123 MYPROG.cbl
```

to set a breakpoint on line 123 of MYPROG.cbl program.

This is a major improvement that enhances the usability of the isCOBOL Debugger in any scenario, from stand-alone debugging (iscrun -d PROGNAME), thin-client debugging (iscclient -d -hostname ip -port n PROGNAME), and remote debugging for all other architectures (iscrun -d -r hostname port)

The Debugger can still load source files from disk if the .class programs are compiled with a previous compiler release for backward compatibility.

In addition, Debugger now has a new integrated View named Console in the isCOBOL IDE that has the ability to attach and de-attach the standard sysout and syserr in its view. When the Console is attached, all the sysout and the syserr output is shown in the new View, using black and red colors respectively. When the Console is de-attached, all the sysout and syserr are shown back in the system console. This results in a clearer view, where the Output View is on the bottom-left corner and shows the results of commands executed from the developer and the single line of code that is executed with "step" command, and the Console view, when attached to activate, is shown in the bottom-right corner and shows the sysout/syserr produced by COBOL program or by the isCOBOL runtime error handling.

In Figure 14, Debugger Console view, the bottom-right portion of the Console view shows the attached sysout/syserr. A pop-up menu is also available to Copy, Select and Clean the content. Figure 15, Debugger Console settings, shows the dialog that appears when selecting the Window menu item Settings – Console, that allows redirection of sysout/syserr which is especially useful in Thin Client, where this change is not dynamic.

Figure 14. Debugger Console view

The screenshot shows the isCOBOL Graphic Debugger interface. The main window title is "isCOBOL Graphic Debugger". The menu bar includes File, Edit, Run, Data, Breakpoint, Settings, and Help. A toolbar with various icons is at the top. The central area displays COBOL source code:

```
2      working-storage section.  
3      77 my-var pic x(3).  
4      77 my-num pic 9(3).  
5      procedure division.  
6      main.  
7          move "abc" to my-var  
8          exhibit named my-var.  
9          move 123 to my-num  
10         display "Value of MY-NUM variable:" my-num upon sysout  
11         if my-num > 100  
12             display "Error, my-num > 100" upon syserr  
13             end-if  
14             goback.
```

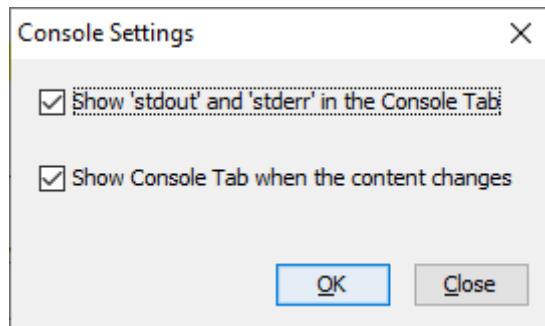
Below the code editor, the status bar shows "line=11 file=deb-console.CBL" and "if my-num > 100". The bottom left pane shows the command history:

```
line=11 file=deb-console.CBL  
    if my-num > 100  
+ MY-NUM = 123  
line=12 file=deb-console.CBL  
    display "Error, my-num > 100" upon syserr  
line=14 file=deb-console.CBL  
    goback.
```

The right side of the interface contains several panes:

- Variables**: A tree view of variables.
- Current Variables**: A table with columns Name, Value, and Hex.
- Watched Variables**: A table with columns Name, Value, and Hex.
- Perform stack**, **Monitors**, **Breakpoints**, **Threads**, **Command history**, and **Console** tabs.
- Console** tab content:  
MY-VAR=abc  
Value of MY-NUM variable:123  
Error, my-num > 100

Figure 15. Debugger Console settings



## c-treeRTG v11.9

A new version of c-treeRTG, V11.9, is available and embedded in the Veryant 2020R2 release. Version v11.9 contains many new features such as TLS/SSL security communication between client and server and other improvements.

Additional details are explained in the installed documentation located in the "c-treeRTG v11.9.0\Server\V11.9.0\_Veryant\_Delivery.pdf" folder.

## TLS/SSL

SSL communication is enabled with a new server configuration setting in the ctsrvr.cfg configuration file, as shown below:

```
SUBSYSTEM COMM_PROTOCOL SSL {  
    SERVER_CERTIFICATE_FILE ctree_ssl.pem  
    SSL_CONNECTIONS_ONLY YES  
}
```

The settings allow the definition of the SSL certificate file, and whether an SSL connection is mandatory (default setting is NO). Additional configuration settings are optionally available.

If a client is configured with a standard c-tree xml client configuration file, the new attributes "ssl" and "sslcert" in the <instance> configuration element must be set, as shown below:

```
<config>  
    <instance ssl="yes" sslcert="ctsrvr.pem" server="FAIRCOMS@hostname"  
        user="admin" password="ADMIN"  
        connect="yes" versioncheck="no">  
    ...  
    </instance>  
</config>
```

If the client is configured using an isCOBOL properties file, the new settings can be configured as shown below:

```
iscobol.file.index.server=FAIRCOMS@hostname  
iscobol.file.index.user=admin  
iscobol.file.index.password=ADMIN  
iscobol.file.index.ssl=true  
iscobol.file.index.sslcert=/path/filename
```

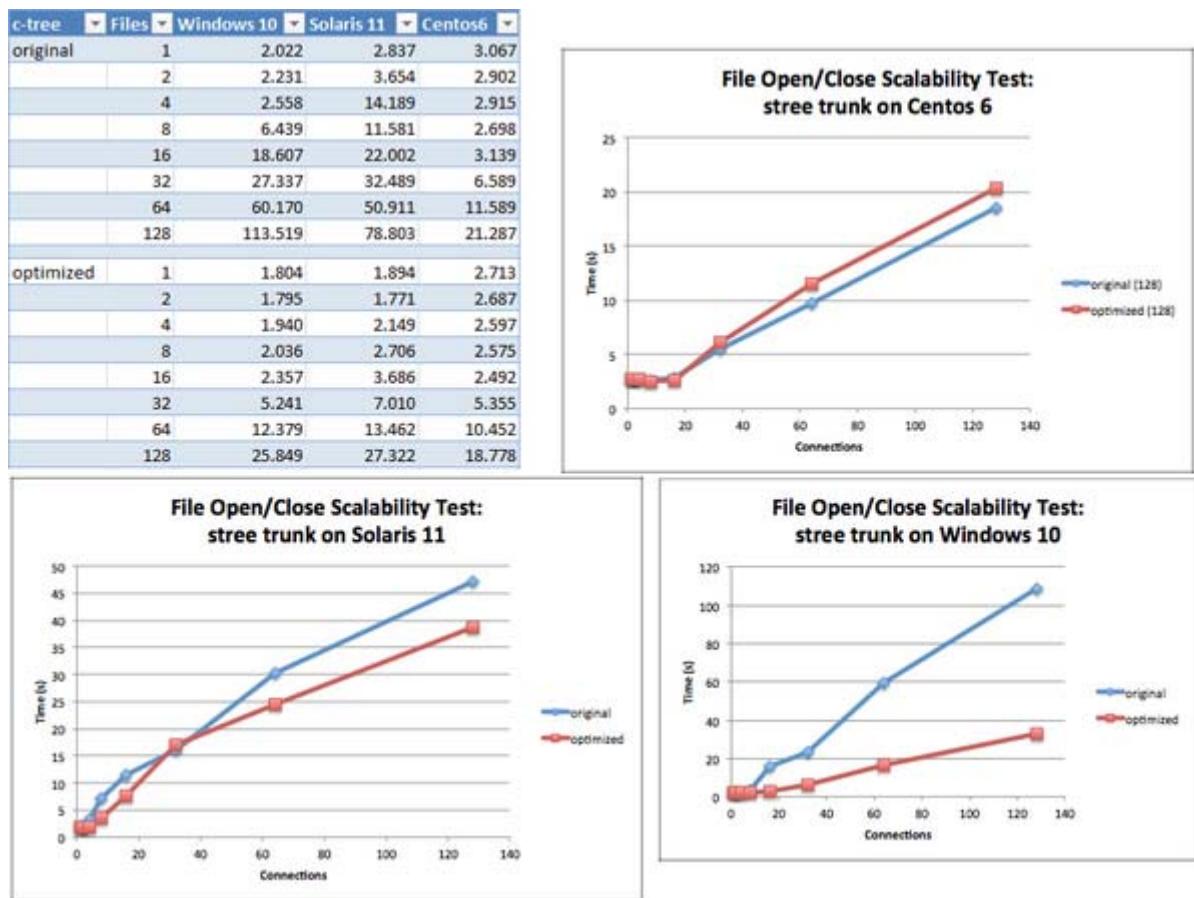
## Other improvements

Additional c-tree client configuration properties have been implemented, such as:

- iscobel.file.index.prefetch.ttl=n to specify how long pre-fetched records should be kept, in seconds
- iscobel.file.index.keycompress.rle=true to enable RLE compression on keys
- iscobel.file.index.keycompress.vlenod=false to restore the legacy LEADING compression on keys
- iscobel.file.index.memoryfile.persist=false to unload memory files when the run unit terminates, instead of when the c-treeRTG server shuts down.

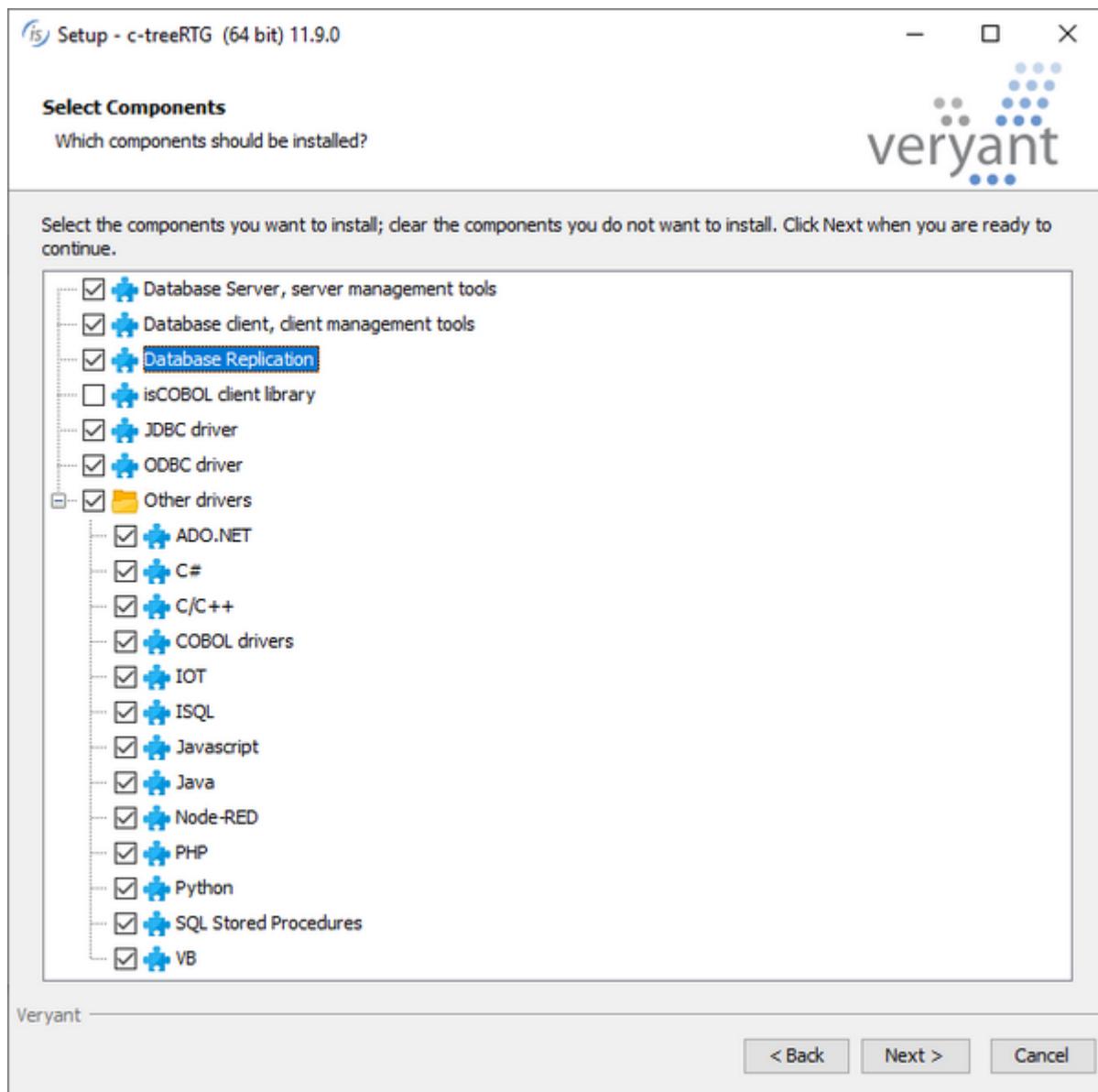
The performance of c-treeRTG Server, when handling concurrent file open and close operations, has been improved, especially when scaling to systems with a large number of CPU cores and concurrent database connections. Figure 16, c-treeRTG scaling test shows scaling test results, with the number of files ranging between 1 and 128, running 2,000 iterations and 128 connections. Result times are in seconds.

Figure 16. c-treeRTG scaling test



The new c-treeRTG Veryant setup, as shown In Figure 16, c-treeRTG 11.9 setup, contains additional components, such as the Replication agent, making it simpler to install, and provides new Drivers with tutorials for additional languages, including C#, C/C++, IOT, ISQL, Javascript, Node-red and VB. This enhances the interoperability of languages that share the same data. To enable these features, you need to replace the embedded license, which only allows access from isCOBOL, with a full license that allow full SQL access and enables the Replication feature.

Figure 17. c-treeRTG 11.9 setup



## Additional improvements

isCOBOL compiler can now be integrated in Apache Ant builds. Ant is a software tool for automating software build processes implemented using Java and is well integrated in software for automation servers like Jenkins. These automation servers help automate the parts of software development related to building, testing, and deploying. Products like Jenkins and Ant are often used together to facilitate continuous integration and continuous delivery (CI/CD).

An integrated system to use isCOBOL from Apache Ant is now available to simplify the usage of isCOBOL compiler in Ant build script.

To use isCOBOL inside Ant script you can now add the following tags to the build configuration file:

```
<taskdef name="iscc" classname="com.iscobol.ant.iscc"/>
```

or use the syntax:

```
<project name="HelloWorld" default="build" basedir=". "
    xmlns:veryant="antlib:com.iscobol.ant">
```

The task name "iscc" provides a task to allow compilation, using the syntax shown below:

```
<!-- Cobol sources -->
<property name="src.dir" location=". " />
<!-- Java classes -->
<property name="build.dir" location="prg" />
...
<iscc
    javacOptions="-classpath ${iscobol-classpath-prop} "
    nosummary="false"
    nowarn="true"
    noerr="true"
    force="true"
    options="-sp../isdef"
    failOnError="true"
    destDir="${build.dir}">
    <fileset dir="${src.dir}">
        <include name="**/*.cbl"/>
    </fileset>
</iscc>
```

Files with a .cbl extension in the current folder and its subfolders are compiled, and the .class will be generated in the "prg" destination folder.

These are the details of supported options in the iscc Ant task:

- **javac**=path to specify the external javac compiler to be used, default is none
- **javacOptions**=options to pass options to the java complier, default is none
- **nosummary**=false to suppress the display of compiler summary information, default is true
- **nowarn**=true to enable or disable compiler warnings, default is false
- **noerr**=true to enable or disable compiler errors, default is false
- **force**=true to force compilation a COBOL program even if it is not out of date, default is false
- **options**=options to set the isCOBOL compiler options used by the iscc command, default is -jc
- **failOnError**=false to continue the build process even if the compiler reports a severe error, default is true
- **destDir**=path to specify the name of the folder where the compiler output is to be written, default is none
- **fileset**=fileset to set the standard Ant fileset, defining a group of COBOL sources to be compiled.  
For details on the syntax see: <https://ant.apache.org/manual/Types/fileset.html>

## Database Bridge

The EDBI routines generated by the Database Bridge functionality when targeting PostgreSQL are now certified for PostgreSQL release 12.

### EIS improvements

A new overload method in `HTTPHandler` class has been added to allow setting the generation of a `DummyRoot` parameter:

```
HTTPHandler::>displayEx(content, hasDummyRoot)
```

The `DummyRoot` is an automatically generated root element added during the creation of the XML or JSON representation of a working storage item, if needed.

### isCOBOL utilities improvements

The isCOBOL Server Panel now supports the Threads View and Thread stack even when the Application Server is executed with a JRE.

The graphical ISMIGRATE wizard can now read isCOBOL configuration files when it starts. This allows you to preload settings to be used in the wizard by using the `iscobol.ismigrate.*` and other relevant properties in configuration files, and referencing the configuration file when starting the wizard. Previously, these variables could only be used when running ISMIGRATE from the command line.

ISMIGRATE has also been expanded to extend existing files. This feature can be enabled from the command line by setting the configuration property:

```
iscobol.ismigrate_open_extend=true
```

## isCOBOL 2020 Release 1 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2020 R1.

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

isCOBOL Code Coverage and isCOBOL Unit Test are enterprise level features that enable developers to write robust test suites and check their effectiveness, allowing the production of a more stable code base in applications.

The 2020R1 release has many new features for GUIs, such as a new control: scroll-pane, and the table-view style for tree-views. These are features that let developers modernize applications.

The EIS suite has been upgraded, enhancing both webClient and webDirect with new features.

Details on these enhancements and updates are included below.

### isCOBOL Code Coverage and isCOBOL Unit Test features

Starting with the 2020R1 Release, isCOBOL is introducing features with enterprise users in mind: isCOBOL Code Coverage and isCOBOL Unit Test. These features have been available in other languages, such as Java, and now they're available to isCOBOL developers as well. They will help developers produce better quality tests for COBOL applications.

These features are now available in the isCOBOL Evolve suite, and can be accessed from the command line or from the Eclipse-based isCOBOL IDE.

isCOBOL Code Coverage will measure the degree to which the source code of a program is executed during test suite runs. A program with high code coverage, measured as a percentage, has had more of its source code executed during testing, which suggests it has a lower chance of containing undetected software bugs compared to a program with low code coverage. This feature can also be used without a real test suite, and it can be enabled via a runtime option. The only requirement is that programs must have been compiled with the -g compiler option.

When running the command:

```
iscrun -coverage IO_PERFORMANCE
```

isCOBOL Code Coverage creates a folder called “htmlReport” which contains an HTML report of the code analysis. The picture below shows the report with a list of all programs executed and the percentage of total and individual program code coverage.

The report shows both the percentage and number of missed statements and paragraphs in each program.

A screenshot of a web browser window titled "Test Coverage". The URL is "file:///C:/Veryant/isCOBOL2020R1/sample/io-performance/htmlReport/index.html". The page displays a table of test results for various programs:

Program	Missed Statements	Cov.	Missed Stmt	Missed Paragraphs	Cov.	Missed	Pars
IO_INDEXED	84.00%	12	75	87.50%	1	8	
IO_LINESEQUENTIAL	97.30%	1	37	100.00%	0	5	
IO_PERFORMANCE	100.00%	0	5	100.00%	0	1	
IO_RELATIVE	98.48%	1	68	100.00%	0	7	
IO_SEQUENTIAL	97.30%	1	37	100.00%	0	5	
PROG_NOT_USED	0.00%	1	1	0.00%	1	1	
Total	16 of 221	92.76%	16	221	2 of 27	92.59%	27

Clicking on the specific program name opens the report related to the single program, highlighting all COBOL paragraphs with the specific percentage of code coverage, as shown in the picture below.

A screenshot of a web browser window titled "IO\_INDEXED". The URL is "file:///C:/Veryant/isCOBOL2020R1/sample/io-performance/htmlReport/IO\_INDEXED.html". The page displays a table of coverage details for the "IO\_INDEXED" program:

Paragraph	Missed Statements	Cov.	Missed Stmt
DELETE_FILE1_TEST	100.00%	0	10
LOAD_FILE1_TEST	100.00%	0	11
MAIN_LOGIC	91.67%	1	12
READ_FILE1_TEST	100.00%	0	13
START_FILE1_TEST	0.00%	10	10
START_TIMER	100.00%	0	3
STOP_TIMER	83.33%	1	6
UPDATE_FILE1_TEST	100.00%	0	10
Total	12 of 75	84.00%	12

Clicking on the source file name shows the corresponding source code, with a green background showing executed code, and a red background showing non-executed statements. A yellow background shows executed conditional statements. All colors are Eclipse's standard for code coverage available for the Java language.

In the picture below, it's easy to understand that the paragraph START-FILE1-TEST was never executed because the IF condition was never satisfied. With this information in mind, better tests can be developed to cover all of the code base.

The screenshot shows a window titled "IO-INDEXED.cbl" in the isCOBOL IDE. The code is color-coded: green for comments and sections like PERFORM, IF, and END-IF; red for sections like OPEN INPUT, CLOSE FILE, and PERFORM START-TIMER; and yellow for sections like MOVE TOTAL-TIME TO TIME-DISP and DISPLAY. A specific section of the code, labeled "START-FILE1-TEST", is highlighted in red, indicating it was not executed during the coverage analysis. The code itself is a series of COBOL statements, including file handling, performance blocks, and output displays.

```
255.      PERFORM READ-FILE1-TEST.
256.      IF TEST-START = 1
257.          PERFORM START-FILE1-TEST
258.      END-IF
259.      PERFORM UPDATE-FILE1-TEST.
260.      PERFORM DELETE-FILE1-TEST.
261.
262.      MOVE TOTAL-TIME TO TIME-DISP
263.      DISPLAY "Total Time: " TIME-DISP
264.
265.      GOBACK
266.
267.
268.      START-FILE1-TEST.
269.      OPEN INPUT FILE1.
270.      PERFORM START-TIMER.
271.      PERFORM VARYING IND FROM 1 BY 1 UNTIL IND IS > NUM-TIMES
272.          MOVE IND TO key-FILE1
273.          START FILE1 KEY IS = key-FILE1
274.      END-PERFORM.
275.      PERFORM STOP-TIMER.
276.      CLOSE FILE1.
277.      ADD TIME-DIFF TO TOTAL-TIME.
278.      MOVE TIME-DIFF TO TIME-DISP
279.      DISPLAY "START: " - TIME-DISP
280.
281.
282.      LOAD-FILE1-TEST.
283.      INITIALIZE D-A01
284.      OPEN OUTPUT FILE1.
285.      PERFORM START-TIMER.
286.      PERFORM VARYING IND FROM 1 BY 1 UNTIL IND > NUM-TIMES
287.      MOVE IND TO KEY-FILE1
```

New configuration settings allow you to customize the report generated by isCOBOL Code Coverage:

- *iscobol.coverage.sessionname=name* sets the name of the coverage session; the default value is the name of the main program
- *iscobol.coverage.sourcefiles=path* sets the list of paths where the COBOL source files are located. If not set, the current folder will be used.
- *iscobol.coverage.classfiles=path* sets the list of paths where the class files are located, and are used by isCOBOL Code Coverage to build the report. If not set, the report will be built using the loaded isCOBOL classes
- *iscobol.coverage.html=path* sets the directory in which isCOBOL Code Coverage will create the report. If not set, the default value is "./htmlReport"
- *iscobol.coverage.xml=path* sets the file path of xml report. If not set, the xml report is not generated. This is needed when using isCOBOL Code Coverage inside the IDE to take advantage of its specific View.
- *iscobol.coverage.analysis.excludes=path* sets the list of isCOBOL classes that will be excluded from the analysis. It can contain the wildcard '\*', for example A\*
- *iscobol.coverage.analysis.includes=path* sets the list of isCOBOL classes that will be included in the analysis. It can contain the wildcard '\*', for example B\*

isCOBOL Unit Test lets developers create automated test suites, which are designed to test that sections of code execute as intended. The more comprehensive the tests included in a suite are, the more stable the resulting application will be. The goal of unit testing is to isolate sections of a program and ensure that they are working correctly.

This feature is activated using a command line option, such as:

```
iscrun -iut -J-ea
```

The –ea Java option is needed to take advantage of the ASSERT statement used in the test programs to show the reason of the failure in the isCOBOL Unit Test report. For example, the following assert statement checks that the condition of a variable named string1 is equal to "my string". If the check fails, the string in the "otherwise" clause will be added in the report:

```
assert string1 = "my string"  
    otherwise "Test string manipulation: Error"
```

If the assert condition is true, the program continues to the next statement. If the entire program is executed without any assert statements failing and no exceptions are raised, it means that the test is successful.

The only mandatory configuration option to set is:

```
iscobol.unit_test.list_file=path
```

where *path* is a single file path or a list of paths that define the list of programs to be executed.

For example, if the test suite needs to execute four programs called TEST1, TEST2, TEST3 and TEST4, the file declared in the iscobol.unit\_test.list\_file needs to contain:

```
TEST1  
TEST2  
TEST3  
TEST4
```

The picture below shows the html report created after running the test suite. In this example, the first 2 programs are executed correctly while TEST3 fails because the above ASSERT statement is executed and the condition is evaluated to false. TEST4 fails because a runtime exception has been raised.

The report will show a list of executed programs, the execution time, and if the tests completed with success. If not, the failed assertions or runtime errors will be included.

The screenshot shows a web browser window titled "test-list-file". The page header reads "isCOBOL Unit Testing" and "test-list-file". Below the header, it says "Executed December 17, 2019 10:55:13 AM". The main content is a table with three columns: "Program", "Time spent (seconds)", and "Test result".

Program	Time spent (seconds)	Test result
TEST1	0.037	Test successfull
TEST2	0.031	Test successfull
TEST3	0.006	<p>Test string manipulation: Error</p> <pre>java.lang.AssertionError: Test string manipulation: Error at TEST3.MAIN(TEST3.java:171) at TEST3.perform(TEST3.java:149) at TEST3.call(TEST3.java:135) at com.iscobol.its.Factory.call(Factory.java:4089) at com.iscobol.its.Factory.call(Factory.java:3962) at com.iscobol.java.IsCobol\$callNoCp(IsCobol.java:124) at com.iscobol.java.IsCobol._call(IsCobol.java:106) at com.iscobol.java.IsCobol.access\$000(IsCobol.java:28) at com.iscobol.java.IsCobol\$1NoExit.run(IsCobol.java:240) at java.lang.Thread.run(Thread.java:748)</pre>
TEST4	0.006	<p>Exception com.iscobol.its.IsCobolRuntimeException: Internal error:com.iscobol.its.CallOverflowException: CALL not found: MYPROG</p> <pre>at com.iscobol.java.IsCobol\$callNoCp(IsCobol.java:142) at com.iscobol.java.IsCobol._call(IsCobol.java:106) at com.iscobol.java.IsCobol.access\$000(IsCobol.java:28) at com.iscobol.java.IsCobol\$1NoExit.run(IsCobol.java:240) at java.lang.Thread.run(Thread.java:748) Caused by: com.iscobol.its.CallOverflowException: CALL not found: MYPROG (java.lang.ClassNotFoundException: MYPROG) at TEST4.MAIN(TEST4.java:162) at TEST4.perform(TEST4.java:142) at TEST4.call(TEST4.java:128) at com.iscobol.its.Factory.call(Factory.java:4089) at com.iscobol.its.Factory.call(Factory.java:3962) at com.iscobol.java.IsCobol\$callNoCp(IsCobol.java:124) ... 4 more</pre>

Total programs 4 with success 2

The Unit Test reports can be customized using the following configuration properties:

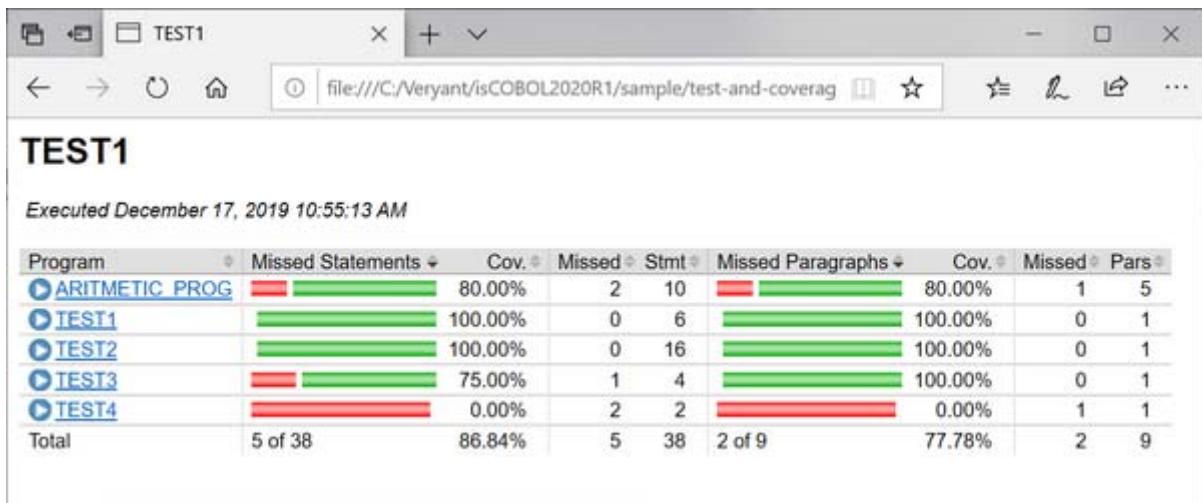
- *iscobol.unit\_test.html=name* sets the directory in which the Unit Test will create the report. Its default value is "./htmlReport"
- *iscobol.unit\_test.xml=path* sets the file path of xml report. If not set, the xml report is not generated. This is necessary when using the Unit Test feature inside IDE to take advantage of its specific View

It's useful to run isCOBOL Unit Test in conjunction with isCOBOL Code Coverage, and both can be activated from the command line:

```
iscrun -coverage -iut -J-ea
```

This can help identify which portions of code still don't have tests associated with them, to make sure the test suites are complete enough to cover all the application code.

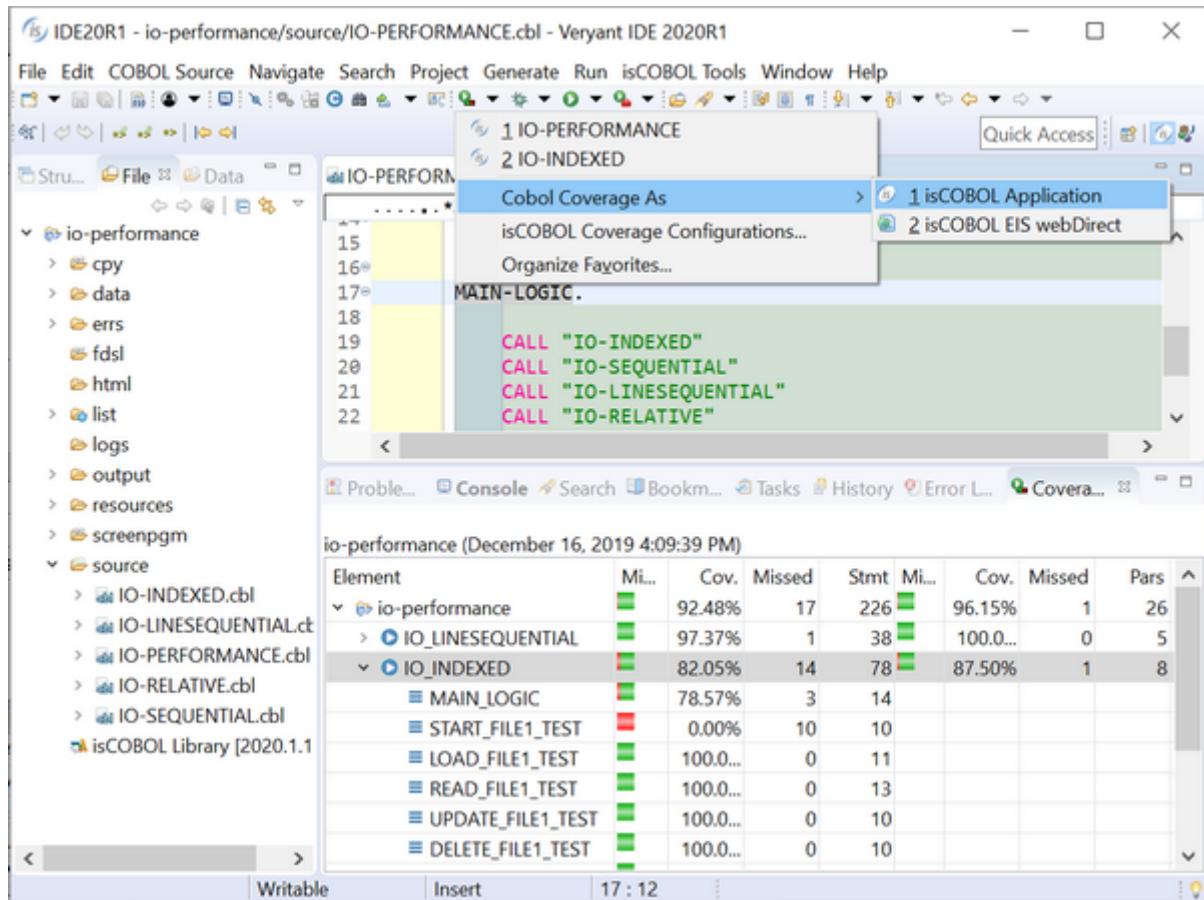
In this scenario the isCOBOL Unit Test report contains a test-list-file link to isCOBOL Code Coverage reports, as shown in the picture below.



## isCOBOL IDE enhancements

In isCOBOL IDE 2020 R1 you can also execute the new Enterprise features in an integrated system, activating dedicated Views to show and analyze the results of isCOBOL Code Coverage and isCOBOL Unit Test.

To execute the isCOBOL Code Coverage feature, as shown in the picture below, the toolbar button "Cobol Coverage As" can be used, and the result of the analysis is shown in the Coverage view. The same menu item is available under the main menu Run and in the source file's context menu.

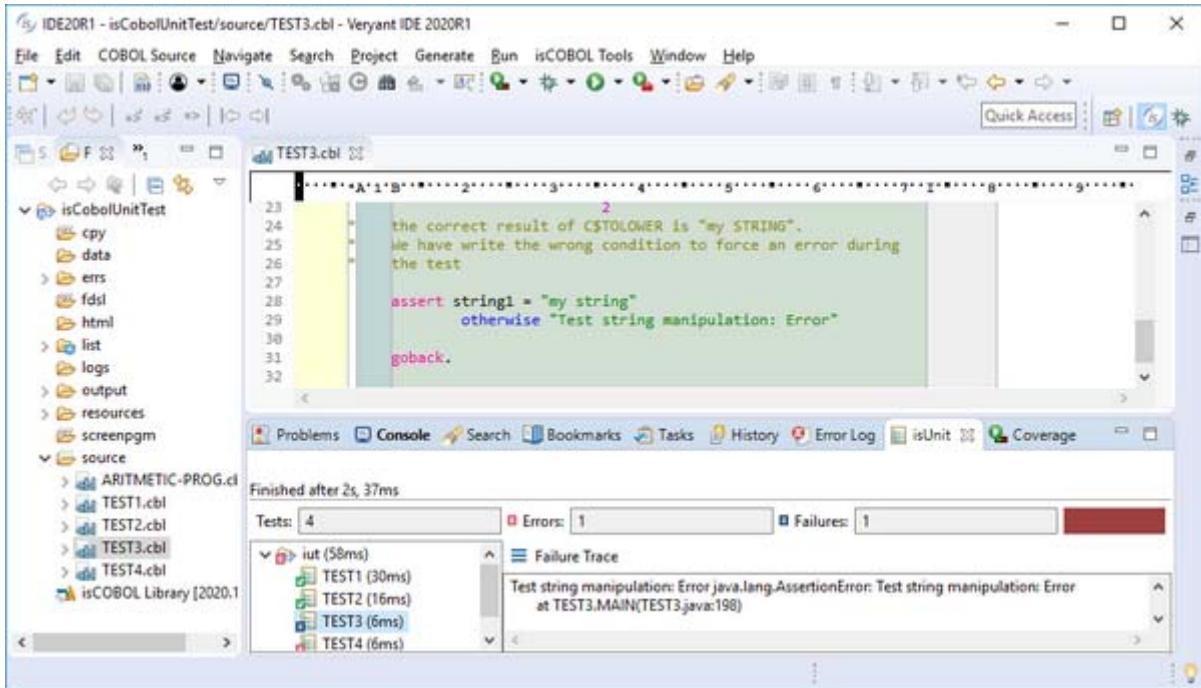


Note that programs must have been compiled with the -g compiler option in order to be processed by the Code Coverage feature.

isCOBOL Unit Test feature can also be accessed from the isCOBOL IDE via a specific run, debug or coverage configuration option. When test execution is completed a specific view will be shown, the isUnit view, as depicted in the picture below. This view shows the list of executed programs, the total number of errors (exceptions) that occurred, the number of failed ASSERT statements and a color bar showing if all tests passed or one or more failed.

Clicking a program in the list displays details relative to that program.

The view is inspired by the standard Eclipse Unit Test view available for the Java language.



The isCOBOL IDE's Code Editor now supports customized code folding. This feature is useful when working with a large source code file, and can help developers hide less relevant pieces of code while concentrating on the important parts.

To enable this feature, the Enable folding option must be set in the *Preferences / isCOBOL / Editor* configuration section, which activates standard folding on sections and paragraphs. To activate a custom folding, select the relevant lines, right-click the folding bar and select the option "Add custom folding". Custom folding sets can be added and removed as needed.

The picture below shows standard folding and two custom folding sets, the first one of which is opened and the second is closed.

The screenshot shows the Veryant IDE 2020R1 interface. The title bar reads "IDE20R1 - iscontrolset/source/ISCONTROLSET.cbl - Veryant IDE 2020R1". The menu bar includes File, Edit, COBOL Source, Navigate, Search, Project, Generate, Run, isCOBOL Tools, Window, and Help. The toolbar has various icons for file operations. The left sidebar shows a project structure with "iscontrolset" expanded, containing subfolders like cpy, data, errs, fDSL, list, logs, output, resources, screenpgm, and source, which further contains files such as CART.cbl, HELPPRG.cbl, ISCONTROLSET.cbl (selected), LOOKUP.cbl, and NOTIFPROG.cbl. The main editor pane displays COBOL source code:

```
*A..B.....1.....2.....3.....4.....5..
951 perform WRITE-DATA-GUI
952 perform DESTROY-RESOURCE
953 .
954
955 DESTROY-RESOURCE.
956     modify h-sta visible 0
957     call "W$MENU" using wmenu-destroy menu-handle
958     call "W$MENU" using wmenu-destroy h-popup
959     call "W$MENU" using wmenu-destroy h-popup-col-grid
960     call "W$MENU" using wmenu-destroy h-popup-grid
961     call "W$MENU" using wmenu-destroy h-tray-menu
962     destroy mask-main ribbon-page-1 ribbon-page-2 h-r
963     call "W$BITMAP" using wbitmap-destroy h-bmpapp
974 .
975
976 LOAD-SET.
977     move 0 to w-tv-item
978     modify tmenu parent      w-tv-item
979             item-to-add   "CONTROL SET"
980             giving w-tv-root
981             hidden-data   a
```

## GUI enhancements

Many improvements to GUI controls have been implemented in this release. A new control, the SCROLL-PANE, is a container that can host other controls. It will display scrollbars as needed to allow the user to pan the visible area.

### Scroll-Pane

IsCOBOL compiler supports this new control and a new property, scroll-group, is available on child components to select their host component.

An example of a scroll-pane is shown below:

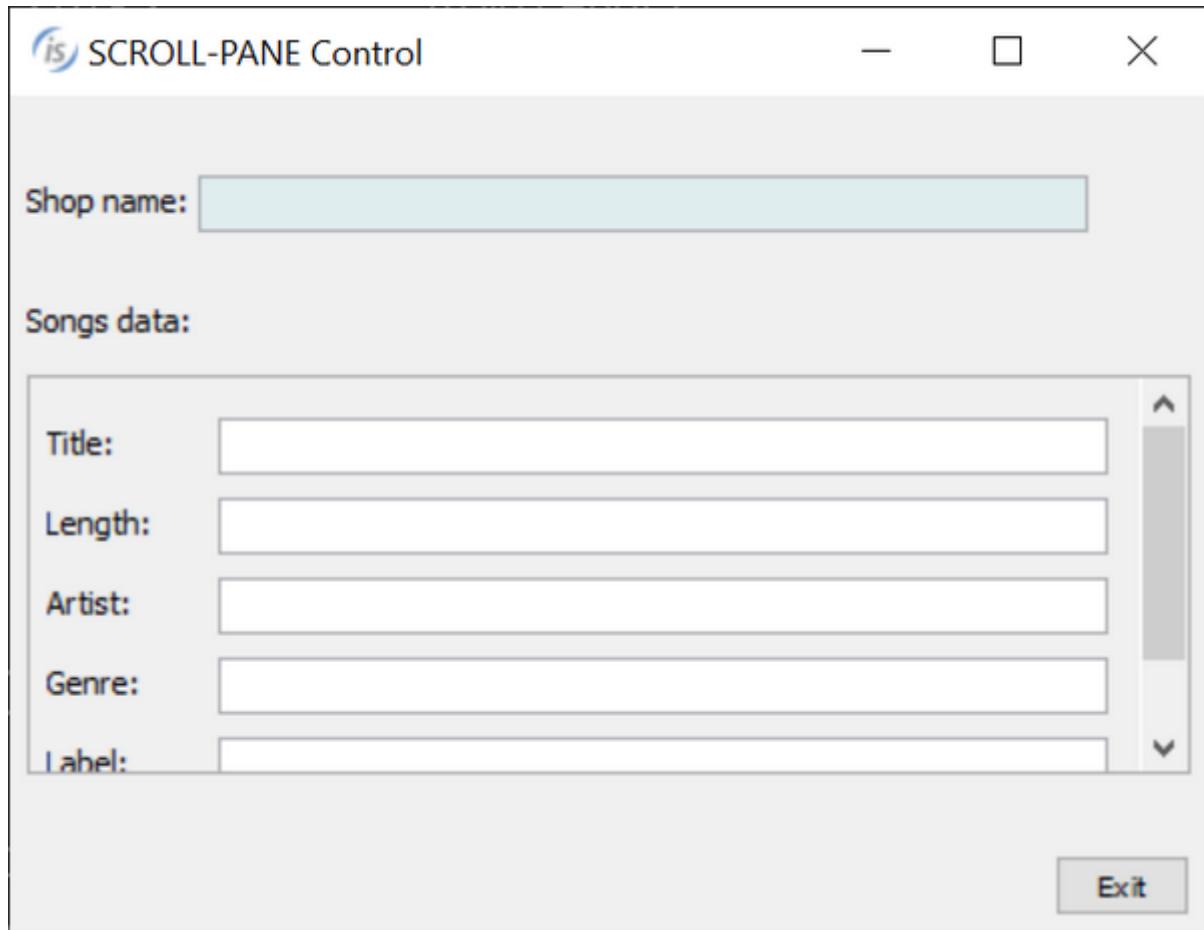
```
03 scroll-pane-1 scroll-pane
    line 8 column 2 size 68 lines 10
    transparent
    border-color rgb x#ACACAC

03 scroll-pane-page scroll-group scroll-pane-1.
05 label
    line 2 col 2 size 8 cells title "Title:"

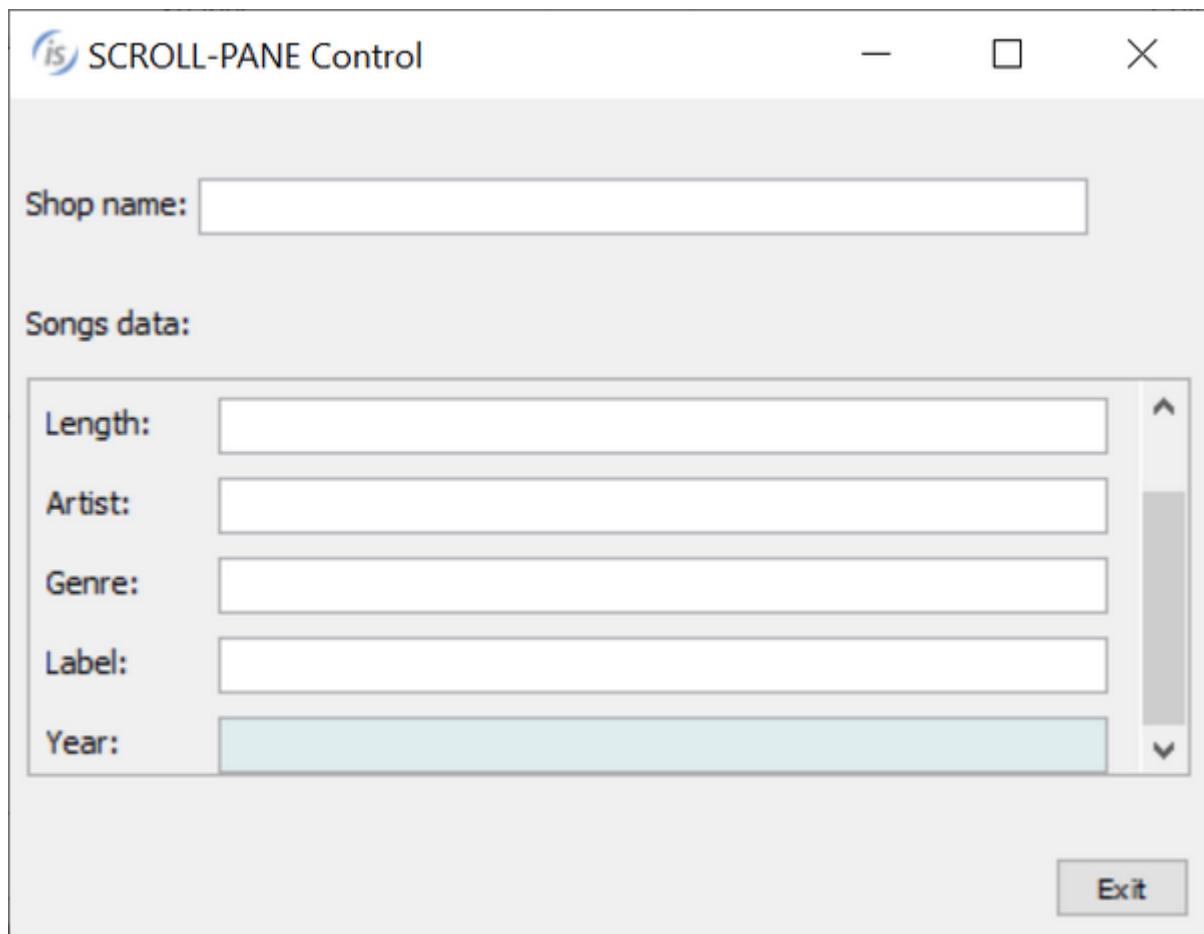
05 entry-field
    line 2 col 12 size 52 cells id 1
```

As focus changes in the child components, the isCOBOL runtime will automatically pan inside the scroll-pane to make sure the focused component is visible. Users can freely use the scroll bars and pan in the content area.

The picture below shows a program running with a scroll-pane container. Initially the upper left portion of the children is displayed. As the user tabs through the controls, the scroll-pane automatically scrolls to show each focused component.



The picture below shows the scroll-pane when the last child has focus.



## Table-View

The tree-view control now supports a new TABLE-VIEW style that activates support for column definitions inside the tree-view. This new style is useful to provide a hierarchical view with tabular data.

The following code snippet declares a tree-view with the table-view style:

```
01 Mask.  
03 tree-table-t tree-view table-view  
    buttons lines-at-root  
    line 2 col 2 lines 15 size 68 cells virtual-width 65  
    display-columns (1, 30, 37, 60)  
    data-columns (record-position of rec-multi  
                  record-position of rec-length  
                  record-position of rec-album  
                  record-position of rec-year)  
    column-headings tiled-headings centered-headings  
    heading-color 257 heading-menu-popup 3 end-color -16774581  
    adjustable-columns reordering-columns  
    event TV-EVT  
.
```

Several properties associated with column management of grid controls are supported in the table-view style, such as display-columns, data-columns, column-headings, and more.

The picture below shows the new style in action.

The screenshot shows a window titled "TREE-VIEW Control - TABLE-VIEW". On the left, there is a tree view with categories like Blues, Blues rock, Latin Rock, Pop, and Rock. Under Blues, there is a node for Eric Clapton with a sub-node for "Next Time You See Her" (Length: 4:02, Album: Slowhand, Year: 1977). On the right, there is a grid view with the following data:

Title	Length	Album	Year
Next Time You See Her	4:02	Slowhand	1977
Maria Maria	4:19	Supernatural	1999
Oye como va	4:36	Abraxas	1970
Lightning in the sky	3:50	Marathon	1979
Foo Foo	6:29	Shaman	2002

At the bottom, it says "You've chosen: Maria Maria" and "Using: mouse". There are "View Source [F2]" and "Exit" buttons.

## Other GUI enhancements

The grid control has been enhanced by adding new properties to manage grids when rows are hidden as a result of active filters, when filterable-column style is set, or because the user is using the automatic search feature, activated by pressing the keyboard shortcut Ctrl-F.

Now the visible rows can be inquired by using the property ROWS-FILTERED, as shown by the following code:

```
inquire my-grid rows-filtered in w-filtered
```

The list of returned rows is the same as with the ROWS-SELECTED property: row1 row2... rowN. If no filter is set on the grid, and there are no hidden rows, inquiring ROWS-FILTERED will return the special value -1.

The property ROWS-SELECTED has been enhanced to allow developers to set a special-value "all" in the modify statement to select all rows, as shown in the code below:

```
modify my-grid rows-selected "all"
```

The new SEARCH-PANEL property has been implemented to force the grid to show the search panel over the column headings. When set to 1, the search panel is always visible even if the grid has the NO-SEARCH style, and the user will not be able to remove it. To enable this behavior, the property can be set in the control's definition, or use the code below:

```
modify my-grid search-panel 1
```

The grid can now be configured to highlight the cell's current row and column heading with a specific color, helping the user identify the current cell content.

Three new properties can be used to configure the colors:

- heading-cursor-background-color
- heading-cursor-foreground-color
- heading-cursor-color

For example, by having the following code set on the screen declaration level:

```
heading-cursor-background-color rgb x#FFDC61
```

the grid will look like popular spreadsheet programs at runtime, as shown in Figure 12, Heading-cursor-color. The heading colors will highlight the current focused cell position.

The screenshot shows a window titled "GRID Control". Inside, a message says "User can change column position and sort data by clicking on the heading". Below this, another message says "User cannot edit the second column". There is a "Search options" button and a "Find Next" button. A table grid displays song data. Row 3, column 3 (the Length column) is highlighted with a yellow background. The cell for song "Help!" in this row is also highlighted with a black border. The table columns are labeled "Title", "Length", "Artist", and "A". The rows are numbered 1 through 7. The "Length" column contains values like "4:03", "2:40", "2:21", etc. The "Artist" column contains "Beatles", "The Rolling Stones", etc. The "A" column contains partial text like "Let It Be", "Revolver", "Help!", etc. At the bottom, there are navigation buttons for rows and columns, and buttons for "View Source [F2]" and "Exit".

	Title	Length	Artist	A
1	Let It Be	4:03	Beatles	Let It Be
2	Yellow Submarine	2:40	Beatles	Revolver
3	Help!	2:21	Beatles	Help!
4	Yesterday	2:07	Beatles	Help!
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

The new NOTIFY-MOUSE style is now supported on all controls, and is used to fire the new events MSG-  
MOUSE-ENTER, MSG-MOUSE-EXIT, MSG-MOUSE-CLICKED, MSG-MOUSE-DBLCLICK.

This allows developers to have finer control on the user interface and the management of mouse events, and provides more freedom in user interface design.

New GUI configurations properties available in the 2020R1 release:

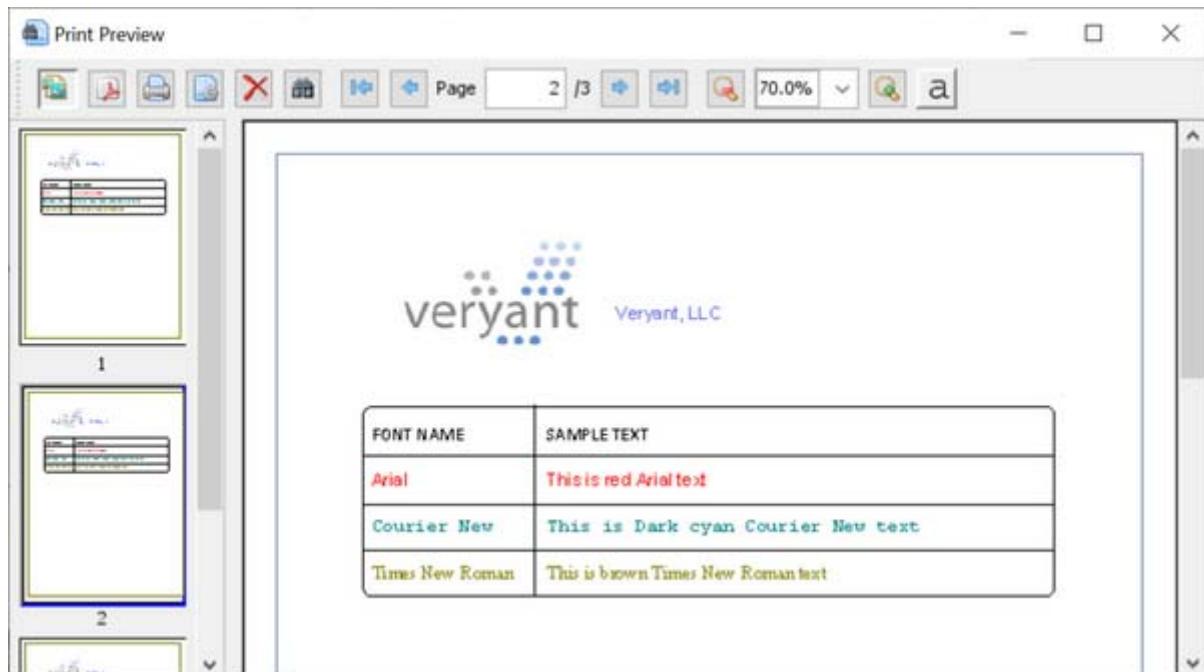
- *iscobol.gui.grid.find\_delay* to set the delay in milliseconds for the Grid search feature
- *iscobol.gui.messagebox.bcolor* to set the default background color of message boxes if not set explicitly on the display statement
- *iscobol.gui.messagebox.fcolor* to set the default foreground color of message boxes if not set explicitly on the display statement
- *iscobol.gui.messagebox.font* to set the default font name and size of text in the message boxes if not set explicitly on the display statement
- *iscobol.gui.windows\_modality* to set the thread blocking behavior of floating windows and message boxes. This is useful on multithread applications.

## Print Preview

The Print Preview dialog will now show page thumbnails, enabled by default, allowing the user to quickly jump to a specific page in a multi-page report. To dynamically control the thumbnails panel, new methods have been implemented in the com.iscobol.rts.print.SpoolPrinter class.

```
void setShowThumbnailsButton(boolean showThumbnailsButton)  
boolean isShowThumbnailsButton()
```

The new print preview with thumbnails is shown in the picture below.



## Database improvements

Performance of data access from Database Bridge to RDBMs tables and c-treeRTG indexed files when logging is enabled has been greatly improved. New configuration properties have been implemented in the isCOBOL runtime to better manage the “InPool” feature of c-treeRTG, which avoids the performance costs of opening an indexed file for the connected client.

### Performance of Database Bridge

Performance of Database Bridge for MySQL has been improved with a new option during the EDBI generation to take advantage of MySQL hints, a feature that allows developers to control the SQL optimizer. This can be set on the edbiis wrapper level with the new –mh option or with the new compiler directive

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

A table of performance gains is shown below, comparing isCOBOL 2019R2 to isCOBOL 2020R1 without and with the new –mh option.

The test was run in Windows 10 64-bit on an Intel Core i5 Processor 4440+ clocked at 3.10 GHz with 8 GB of RAM, using Oracle JDK 1.8.0\_231 and MySQL 5.6. All times are in seconds. The COBOL program executes the START on different keys, then executes READ NEXT for all the records in a table containing 200,000 records.

The EDBI routine uses light cursors (-dmld wrapper option or compiler configuration *iscobol.compiler.easydb.light\_cursors=2*), causing a new cursor to be opened every 100 records. The use of hints saves the time spent by the MySQL query optimizer to choose which index to use to scan the table.

operation	2019 R2	2020 R1	2020 R1 with -mh
reading records after START on Primary Key	7,51	7,29	4,81
reading records after START on Alternate Key	11,05	10,82	5,53

A new Database Bridge configuration option has been added to generate a smarter START statement that executes less query instructions:

```
iscobol.easydb.limit_dropdown=n
```

where *n* can be:

- |   |  |
|---|--|
| 0 | off, the default behavior  |
| 1 | partial, to activate the optimization on START with the WITH SIZE clause |
| 2 | full, to activate the optimization on START without WITH SIZE clause     |
| 3 | all, to activate the optimization on any START statement                 |

As shown in the picture below, a performance comparison is made between isCOBOL 2019R2 and isCOBOL 2020R1 without and with the new configuration. The test was run in Windows 10 64-bit on an Intel Core i5 Processor 3210M 2.50 GHz with 16 GB of RAM, using Oracle JDK 1.8.0\_231 and Oracle 11 Database server. All times are in seconds. The COBOL program executes the START statement followed by a loop of READ NEXT to read the records that satisfy the conditions set on segments of the key. The table contains 100,000 records.

operation	2019 R2	2020 R1	2020 R1
configuration	iscobol.easydb.limit_dropdown=3		
START on Key equal	34,04	33,06	18,15
START on Key not less	34,13	33,39	18,10

## Performance of c-treeRTG indexed files

A new c-tree server configuration variable, DELAYED\_DURABILITY, has been implemented in the embedded isCOBOL 2020R1 c-treeRTG version to increase performance of c-treeRTG indexed files when the logging feature is active.

Enabling logging is beneficial for a number of reasons:

- Safety: in case of file corruption caused by events such as hardware failure or unexpected power loss, when the c-treeRTG server is restarted with logging enabled, files are rebuilt automatically, ensuring data integrity, and is user-transparent.
- Replication: c-treeRTG replication requires logging to be enabled, since the replication engine relies on logging information to properly update all the servers involved. Replication can be one-directional, where data saved on a master server is replicated to a backup server, or multi-directional, where two or more servers continuously update each other in real-time. Replication has a great benefit: failover, since if a server goes offline, any other server in the replication cluster can fulfill requests for the missing server.

Since logging could slightly impact performance and increase disk usage, it is possible to enable it on only specific, highly modified files, while other less important files such as lookup tables can be used without logging to maximize performance.

The following command can be used to activate logging on a specific file:

```
ctutil -tron filename
```

All COBOL statements that update records (such as WRITE, REWRITE, DELETE) have been improved, as shown in the picture below, a performance comparison between isCOBOL 2019R2 and isCOBOL 2020R1 without and with the new configuration variable. The test was run in Windows 10 64-bit on an Intel Core i5 Processor 3210M 2.50 GHz with 16 GB of RAM, using Oracle JDK 1.8.0\_231. All times are in seconds. The indexed file contains 300,000 records.

The OPEN-READ-CLOSE test showcases the performance gains that can be obtained using the InPool feature of c-treeRTG. The test program performs a loop consisting of a single READ inside OPEN and CLOSE statements. The configuration options that handle pooling allow developers to flexibly configure which files can use the InPool feature. These are:

```
iscobol.file.index.filepool=true
iscobol.file.index.inpool=true
iscobol.file.index.filepool.size=n
iscobol.file.index.#.inpool=true
```

isCOBOL operation	2019 R2	2020 R1	2020 R1
Ctree version:	11.6.0.64778	11.6.0.65002	11.6.0.65002
server configuration		<b><i>DELAYED_DURABILITY 1</i></b>	
WRITE	97,28	96,06	51,86
READ	3,75	3,73	3,71
REWRITE	55,88	53,84	11,65
DELETE	86,09	85,75	42,93
Total	243,00	239,38	<b>110,15</b>
client configuration		<b><i>file.index.filepool=1</i></b> <b><i>file.index.inpool=1</i></b>	
OPEN READ CLOSE	11,73	11,21	<b>1,01</b>

## isCOBOL Compiler

Starting from isCOBOL 2020R1, the compiler supports the new POSITIONAL clause in the MOVE statement to easily move dynamic variables (X ANY LENGTH, OCCURS DYNAMIC,...) used inside structures into other structures without dynamic variables and vice versa.

This proves useful, for example, when moving data from FD declaration, which has static content, to WORKING-STORAGE items, which can contain dynamic variables, to minimize memory usage.

Intrinsic functions execution has been enhanced, and a new EFD directive has been added for temporary tables.

### Move Positional

The new POSITIONAL clause is similar to the existing CORRESPONDING clause, but instead of relying on variable names to identify matching items, it will use the positional order of items to match and move data between structures. The first item in the source structure will be moved to the first item in the target structure, the second to the second, and so on.

For example, the following code snippet defines 2 structures that are comparable in the field positioning but one structure contains only fixed data items, while the other contains dynamic data items:

```
WORKING-STORAGE SECTION.  
01 struct-fix.  
  03 g1-name          pic x(50).  
  03 g1-table         occurs 100.  
    05 g1-account-id   pic 9(3)      value 0.  
    05 g1-account-short-des pic x(20)    value space.  
    05 g1-account-notes  pic x(1000)   value space.  
  03 g1-address       pic x(50).  
01 struct-dyn.  
  03 g2-name          pic x any length.  
  03 g2-table         occurs dynamic  
                capacity cap-g2-table-occ  
                initialized.  
    05 g2-account-id   pic 9(3)      value 0.  
    05 g2-account-short-des pic x(20)    value space.  
    05 g2-account-short-des pic x any length value space.  
  03 g2-address       pic x any length.
```

The new POSITIONAL clause can be used as shown in the statement:

```
move positional struct-fix to struct-dyn
```

and the compiler will translate that one statement in the following moves:

```
move g1-name to g2-name  
perform varying ind from 1 by 1 until ind > 100  
  move g1-account-id(ind)      to g2-account-id(ind)  
  move g1-account-short-des(ind) to g2-account-short-des(ind)  
  move g1-account-notes(ind)    to g2-account-notes(ind)  
end-perform  
move g2-address to g2-address
```

This will make maintenance of COBOL code with large data structures much simpler.

Moreover, the DELIMITED BY DEFAULT VALUE clause can be used as in the following statement:

```
move positional struct-fix to struct-dyn delimited by default value
```

and the compiler will translate it as:

```
move g1-name to g2-name
perform varying ind from 1 by 1 until ind > 100
  if g1-account-id(ind)      = 0 and
    g1-account-short-des(ind) = spaces and
    g1-account-short-des(ind) = spaces
    exit perform
  else
    move g1-account-id(ind)      to g2-account-id(ind)
    move g1-account-short-des(ind) to g2-account-short-des(ind)
    move g1-account-notes(ind)   to g2-account-notes(ind)
  end-if
end-perform
move g1-address to g2-address
```

This way, dynamic occurs variables will be filled until default values are found, reducing memory usage to what is actually being used.

The POSITIONAL clause can also be used when moving two fixed structures with different children and different pictures, since it relies on positional ordering to execute the moves.

## Intrinsic functions

The compiler now supports a shorthand syntax to invoke functions using the \$ symbol in place of the “function” keyword. This simplifies the way to execute intrinsic functions and could also be useful for compatibility with other COBOLs.

For example, the syntax:

```
display $length(g2-address)
display $capacity(g2-table)
```

is the equivalent of:

```
display function length(g2-address)
display function capacity(g2-table)
```

The two functions length and capacity will be executed during the DISPLAY statement.

Intrinsic functions can be used in any COBOL statement that requires a data item, such as MOVE, STRING, or any conditional statement, such as IF, EVALUATE, PERFORM UNITL.

## EFD TEMPORARY directive

The compiler now supports a new EFD directive to mark a table as TEMPORARY instead of PERMANENT. This directive is used by the Database Bridge product. Temporary tables are useful as “working files”. They are usually needed to save temporary data that is not being used concurrently and does not need to be stored permanently. Usually these temporary tables are kept in memory to provide maximum performance.

The following code snippet shows how to define the table “mywork” as temporary:

```
$efd temporary
  fd mywork.
  01 mywork-rec.
    03 mywork-k    pic 9.
  ...
```

The compiler will natively support temporary tables for any RDBMS that supports them.

Oracle 18c supports two kinds of temporary tables: global and private, and those can be specified using the global and private values in the temporary directive. Unlike traditional database servers, global temporary tables in Oracle are permanent database objects that store data on disk and objects are visible to all sessions, but data written in the tables is only visible to the session that created it. At the end of the session the data is removed, but the table remains. Private temporary tables, on the other hand, are memory-based tables, only visible to the session that created it, and are automatically dropped at the end of the session or transaction.

Example of the \$efd temporary directive for Oracle 18c:

```
$efd temporary = global
```

or

```
$efd temporary = private
```

## isCOBOL Debugger

Starting from isCOBOL 2020R1 the isCOBOL Debugger, used to debug ThinClient applications, makes it easier for multiple developers to debug applications running on the same Application Server. To allow this, the Application Server now spawns a separate process for each ThinClient application launched with the -d option, allowing isolation of processes so that debugging an application will not have influence on other applications that may be running in the same Application Server.

This allows debugging of applications on production servers without using a separate Application Server running on a different port number.

By default, port numbers from 9999 to 10099 are used, allowing a maximum of 100 concurrent debugging sessions. The range of ports can be configured using the new configuration variable `iscobol.as.debugport_range`. For example, setting the property using:

```
iscobol.as.debugport_range=20010-20013,20050
```

the Application Server will allocate ports 20010, 20011, 20012, 20013, 20050 for debugging sessions.

Using the new configuration property is transparent to the client, and there is no need to specify a port to connect to for debugging, as the server will handle it automatically. The same command line can be used to start multiple debugging sessions from multiple users, such as:

```
isclient -hostname ip-server -port 10999 -d PROGNAME
```

For compatibility, the previous command syntax forcing a specific debug port is still supported:

```
isclient -hostname ip-server -port 10999 -debugport 20012 -d PROGNAME
```

Using the isCOBOL 2020R1 release, we suggest removing the `debugport` option from the command line, and letting the server choose the first available debug port.

If bypassing the new Application Server behavior is needed, the following configuration variable can be set to fall back to the previous multithreaded mode:

```
iscobol.as.multitasking=0
```

## isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. In isCOBOL 2020R1, there are many updates to webDirect and webClient. Also, the HttpClient class can now consume existing web services with PUT and DELETE requests.

### webDirect enhancements

webDirect, Veryant's technology that allows programs with screen sections to run in a web browser by converting them to HTML/CSS/Javascript equivalents, has been updated with several enhancements.

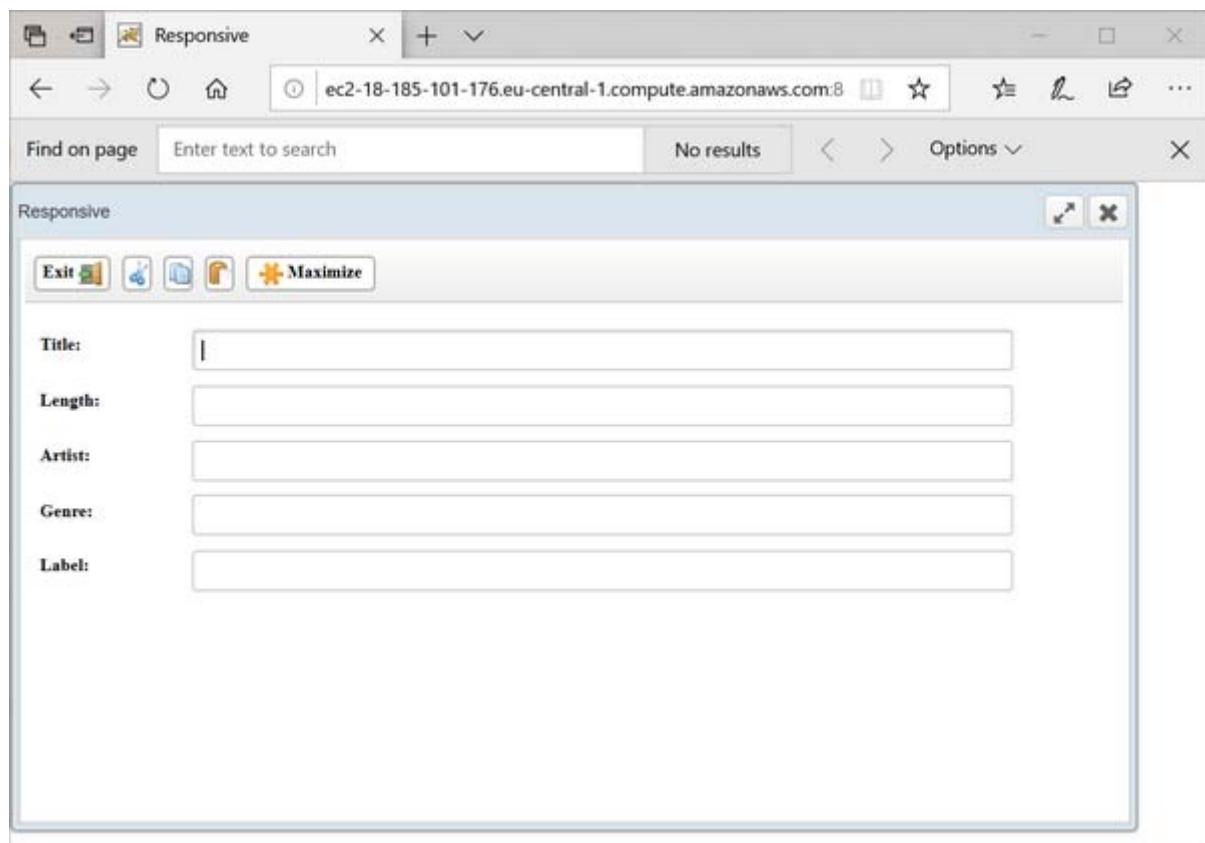
The grid control now allows keyboard navigation as its desktop counterpart, and supports inquiring the ROW-CAPACITY property.

A new configuration property has been added to allow numeric input field to display a numeric keypad on mobile devices:

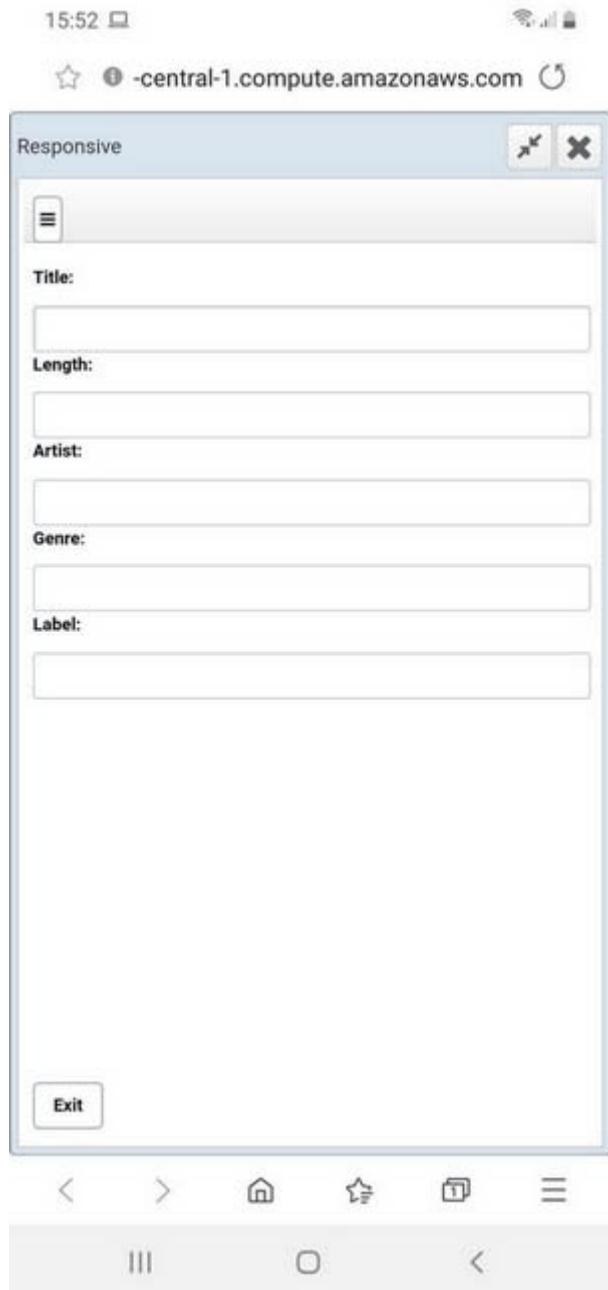
```
iscobol.wd2.mobile_numpad=1
```

All layout-managers are now fully supported in WebDirect, including the "Im-responsive" layout manager. This allows developers to target multiple devices with different screen sizes and resolutions using WebDirect without having to use HTML/CSS implementation or WebClient, and to have the application respond properly when the user resizes the application or browser windows.

The picture below shows the user interface of a program running on a desktop system, where each entry field is displayed next to it, since screen real estate allows it.



The picture below shows the same application running on a smartphone screen, where labels are displayed above the entry fields to optimize the limited width of the screen.



## WebClient enhancements

WebClient is Veryant's solution for running desktop applications on a remote server and interacting with it through a web browser. To interact with the application on devices that do not have a physical keyboard, WebClient provides a "Keyboard" soft button that displays the device's virtual keyboard when pressed. The keyboard now can also be activated by double tapping on the screen. A new application configuration field has been added, "Minimum display width in pixel for keyboard button". This allows the developer to set a minimum display resolution for the keyboard button. When the display size is less than this setting the Keyboard button is hidden, leaving all the screen real estate available to the application, and minimizing the chance of the "Keyboard" button covering the user interface elements.

Additionally, IME keyboards are now fully supported, extending support for languages such as Chinese, Japanese and Korean.

## HTTPClient

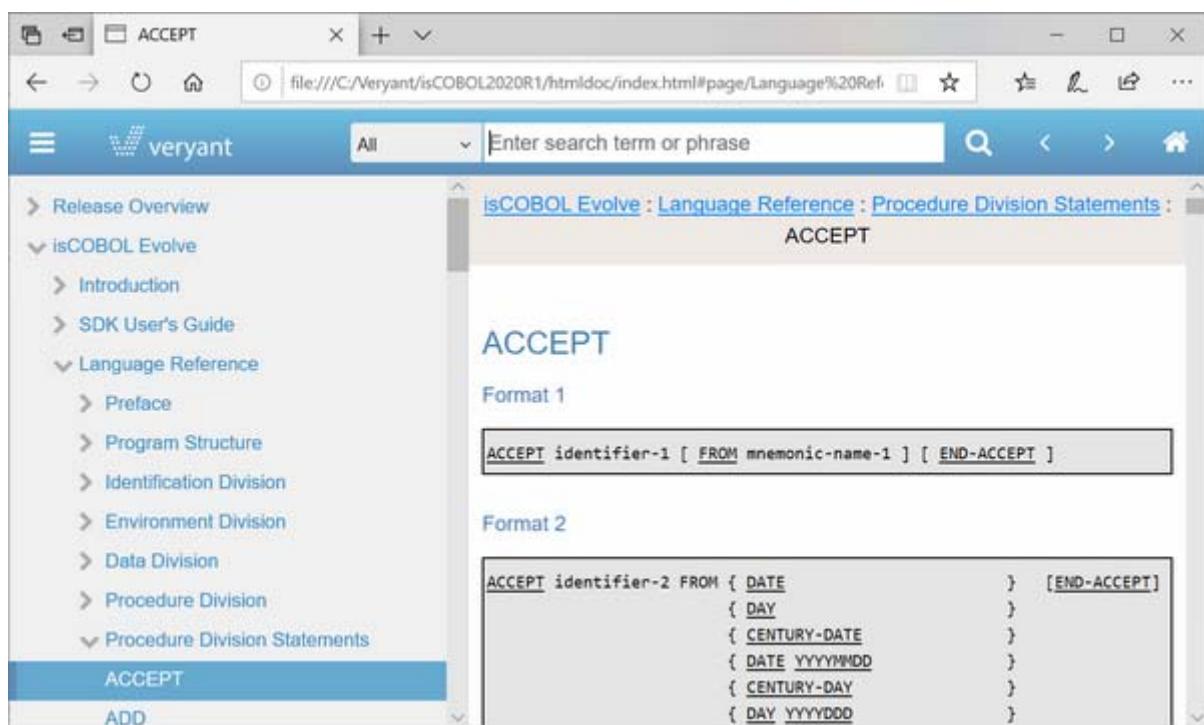
HTTPClient is a class that allows COBOL programs to interact with Web Services. It has been updated to manage PUT and DELETE requests, in addition to the already implemented GET and POST requests.

The new method signatures are shown below:

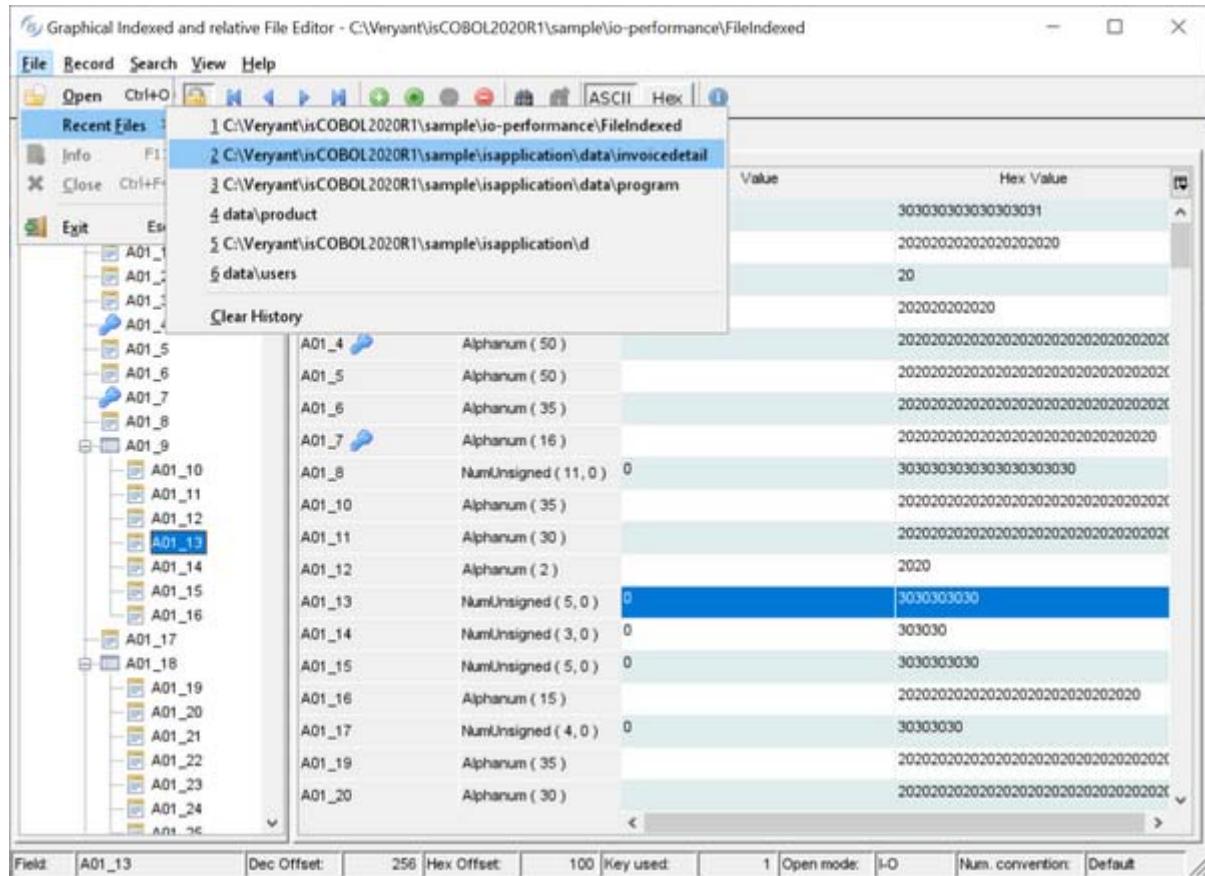
```
public void doPut(strUrl)
public void doPut(strUrl, params)
public void doPutEx(strUrl, content)
public void doPutEx(strUrl, type, content)
public void doPutEx(strUrl, type, content, hasDummyRoot)
public void doDelete(strUrl)
public void doDelete(strUrl, params)
```

## Additional improvements

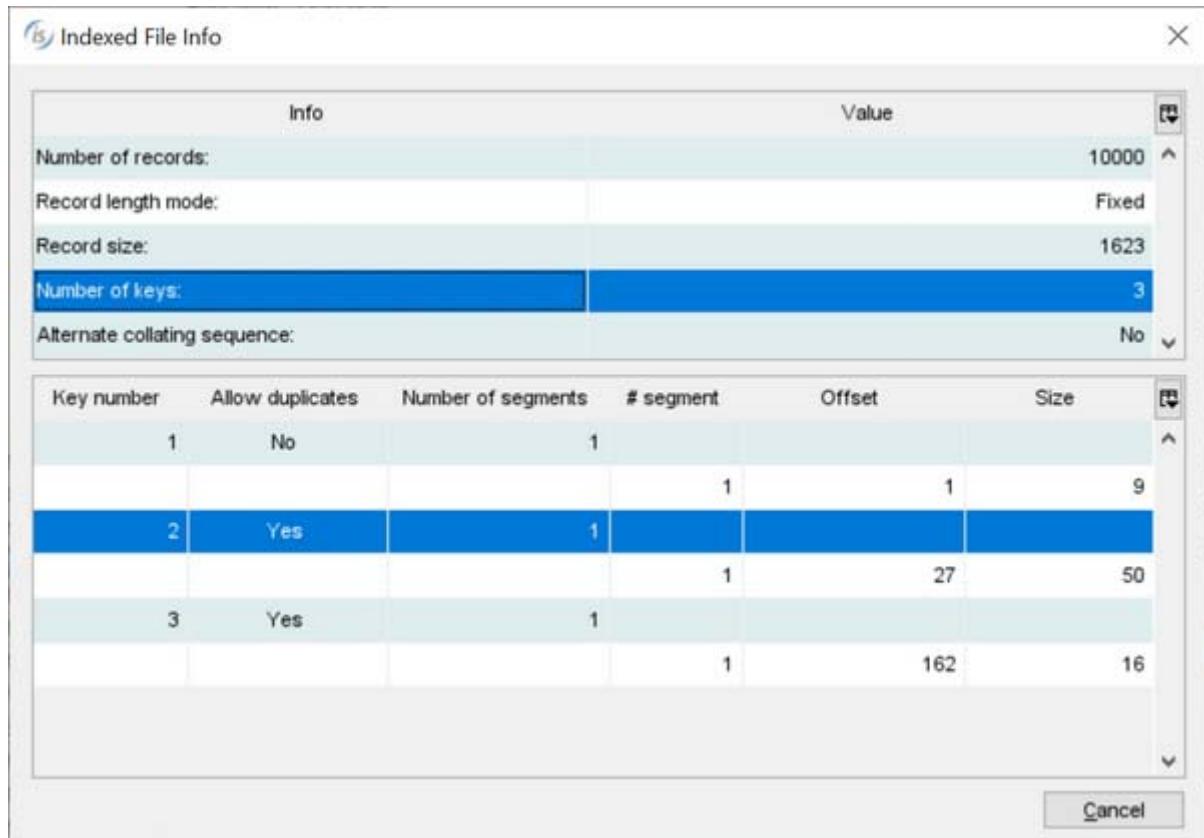
Since devices are becoming more and more diverse, the isCOBOL Documentation has been redesigned with a new modern look and feel and is now fully responsive, allowing it to be accessible to developers from mobile devices.



GIFE, the Graphical Indexed and relative File Editor, now manages recently opened files, making it simpler to reopen with recently used settings, such as Open Mode and Efd file. The picture below shows the new recent opened file menu items, allowing the developer to select a file with a single click. The recently used file is saved in the \$user/gife.properties file when the user exits the GIFE utility.



The GIFE's File info dialog displays the file record size, collation and key definitions.



## isCOBOL 2019 Release 2 Overview

### Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2019 R2.

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

The 2019R2 release has many new features for GUIs, such as a new zoom layout-manager, new borders and color management, and features that let developers modernize applications with little or no effort.

isCOBOL 2019R2 has increased performance for CALL statements and C-Tree file handling.

The EIS suite has been upgraded, giving more power to developers and more control over application deployed with webClient.

Details on these enhancements and updates are included below.

### GUI enhancements

Many improvements to GUI controls have been implemented in this release, such as a new Layout Manager to easily manage scaling GUIs, custom borders that can now be set on controls, more color choices to enrich GUIs, and a new hook for mouseovers..

#### Zoom Layout Manager

Starting with isCOBOL 2019R2, all the isCOBOL GUI, Graphical windows, Screen Programs or isCOBOL WOW programs generated by IDE can take advantage of an easy to use and low impact layout manager to handle application resizing.

By setting the configuration setting:

```
iscobol.gui.layout_manager=lm-zoom
```

the new Zoom Layout manager is activated, windows automatically become resizable, and all controls are adjusted in size when increasing the window width, and in font size when increasing the window height. This behavior is completely automatic, requiring no effort from developers. All that is needed to enable this behavior is to set a configuration property.

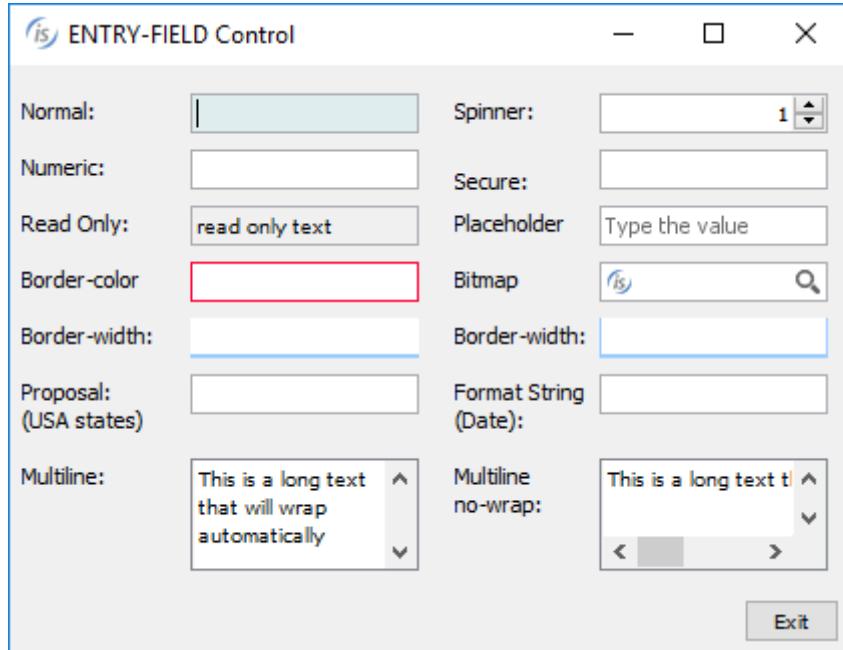
This helps quickly and easily to solve the resizing issues of running applications on a variety of monitors with different sizes and resolutions, with zero code changes.

Individual windows can be targeted by enabling the LM-ZOOM layout manager in the display window statement, as shown in the code below.

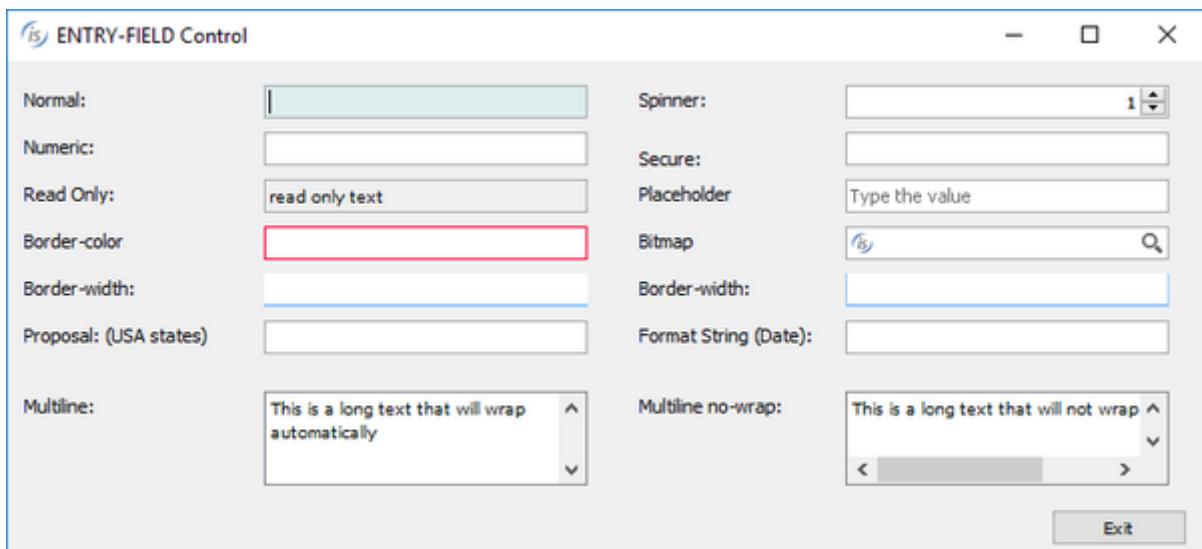
```
77 zoom-layout handle of layout-manager, lm-zoom.  
display standard graphical window resizable  
layout-manager zoom-layout
```

When migrating programs from COBOL-WOW, the LM-ZOOM Layout Manager can be enabled on specific forms by setting the new layoutManager property in the isCOBOL IDE WOW Painter.

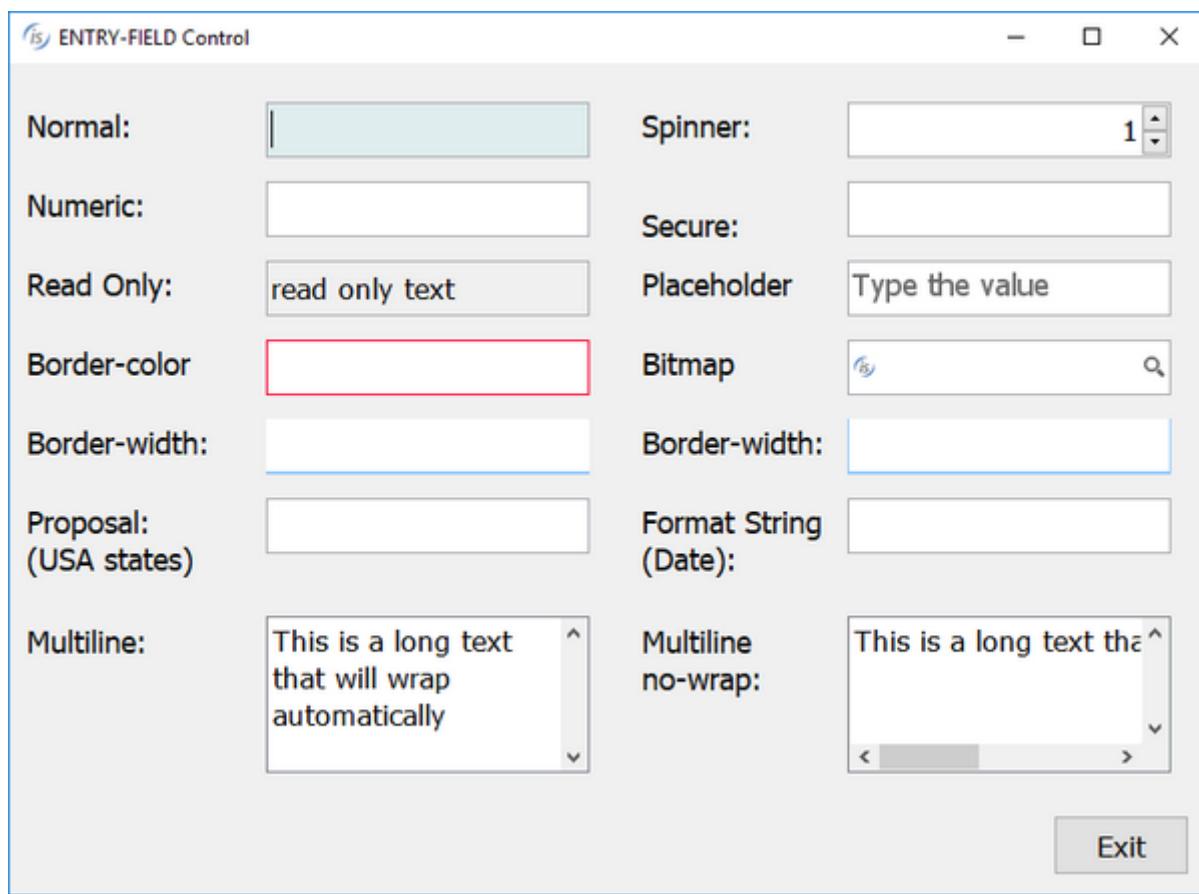
The picture below shows how the program runs at startup, before the user resizes the window.



The picture below shows how the GUI looks like after the user stretches the window horizontally.



The picture below shows how the GUI reacts after the window is resized both horizontally and vertically.



## Custom borders

Controls have always had a `border-color` property that allows custom border colors to be applied. A new property, `border-width`, now allows you to control how borders are created on selected controls. The property is an array of 4 integers that specify the width in pixels of the top, left, bottom and right side of the control.

For example, by setting the following property of an entry-field

```
border-width (2, 2, 2, 2)
```

the control will have 2 pixels-width borders on all sides. Values can be combined to give the application a more modern look, or to make specific controls stand out from others. For example, by setting the following borders on an entry-field

```
border-width (0, 0, 1, 0)
```

only the bottom border will be activated, resulting in a control that looks much like Google's material design guidelines input elements.

Using the values:

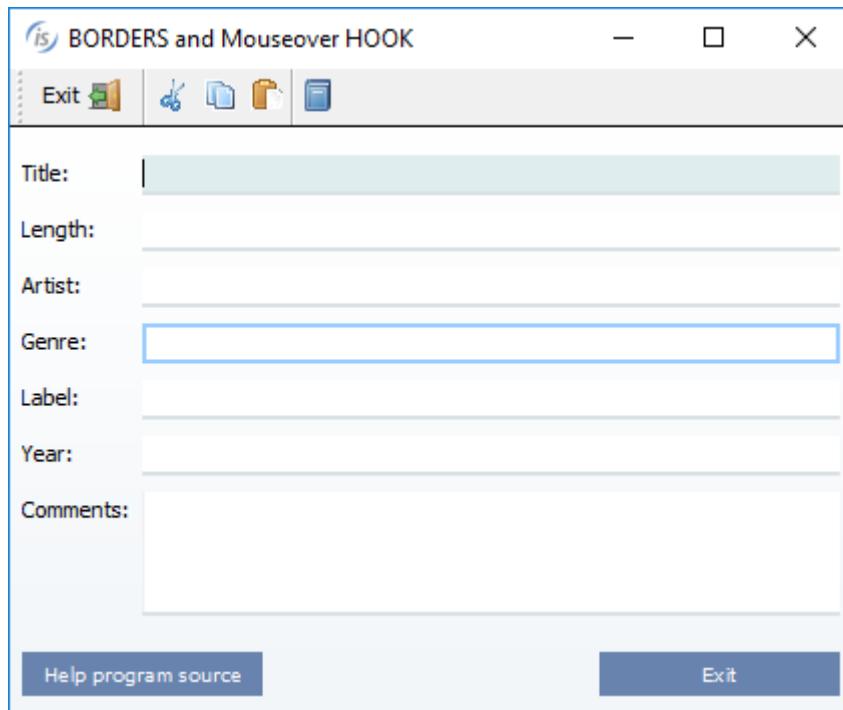
```
border-width (0, 1, 1, 1)
```

a “basket-like” effect can be achieved.

These styles can be combined with new additional configuration settings to generate different border-widths and have them applied automatically when controls gain focus (*iscobol.gui.curr\_border\_width*) or when the mouse hovers over controls (*iscobol.gui.rollover\_border\_width*).

See more details in the [New configuration properties](#) paragraph in the Framework improvements section below.

The picture below shows a modern looking GUI using the border-width property to set the default border, the currently focused control border, and the border style for controls being hovered by the mouse.



## Additional color settings

New properties have been added to set different colors for specific items or the selected item in controls that handle items, such as tree-views, list-boxes and combo-boxes. Combining all the new properties will ease the effort of modernizing existing GUI applications.

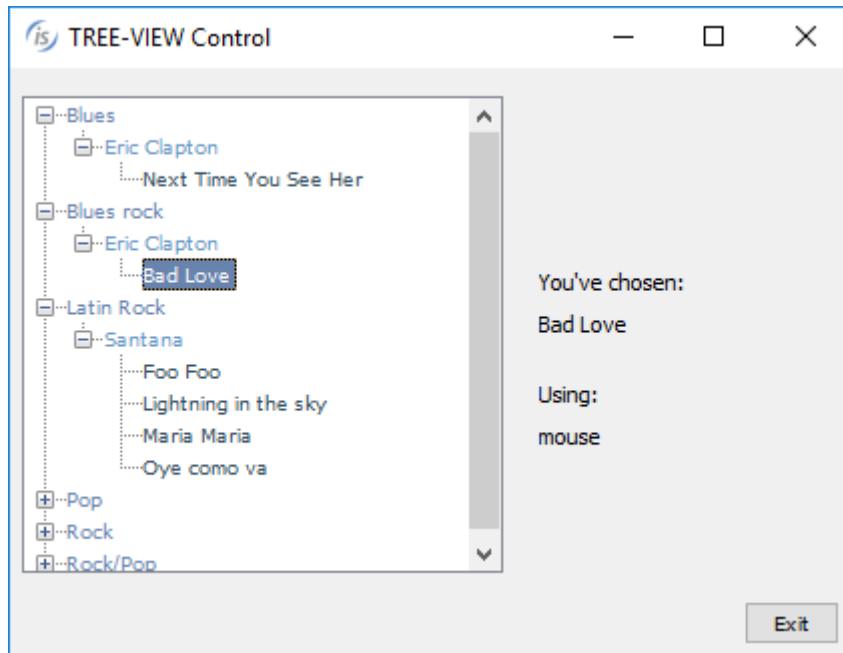
The code snippet below shows how to set a specific color for the selected item in the tree-view declared in the screen section:

```
03 Tv1 tree-view  
    selection-color 483  
    ...
```

The code snippet below shows how to set colors on a specific item added to a tree-view using code run at runtime:

```
modify Tv1 item-to-add "item1"  
    item-color 5  
modify Tv1 item-to-add "item2"  
    item-foreground-color rgb x#FF0000  
    item-background-color rgb x#84FFFF
```

The picture below shows how to enhance the looks for tree-views using different colors on different item levels.



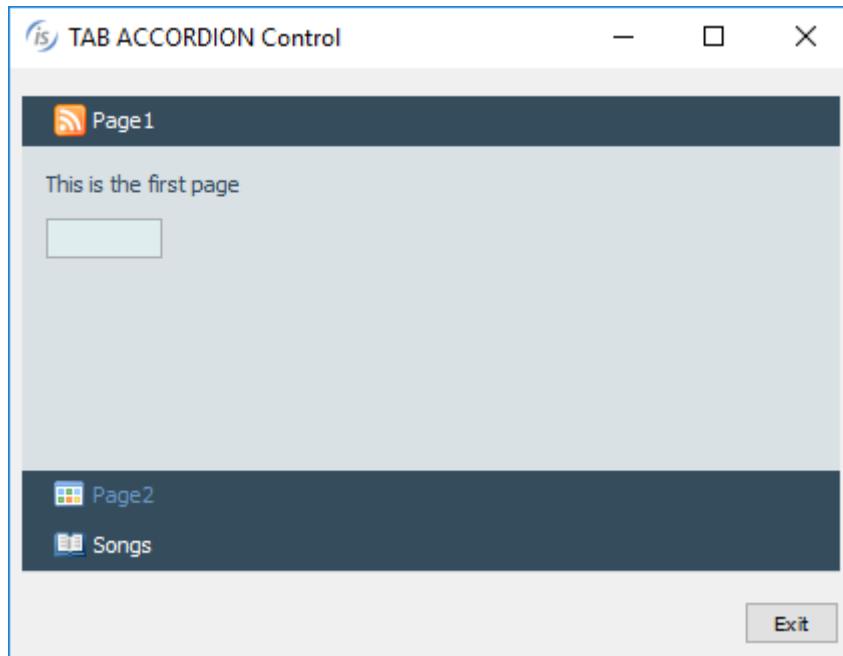
Accordion tab-controls also support the new properties and styles to enhance the color selection, allowing a specific color to be set for the title tag area and the color to be applied when the mouse hovers over the control. Additionally, the new style *tab-flat* will cause the tab title to be rendered with a flat style.

The *tab-delay* property allows you to set the duration of the animation effect displayed when the user selects a different tab, resulting in a smooth transition effect.

The code snippet below configures additional colors in the accordion declaration in screen section:

```
03 Acc1 tab-control accordion
  tab-flat
  background-color rgb 145
  foreground-color rgb 10745433
  tab-background-color rgb 145
  tab-foreground-color rgb 10745433
  tab-rollover-color rgb 16711680
  tab-delay 2000
  ...
```

The picture below shows the effect of setting colors with the tab-flat style.

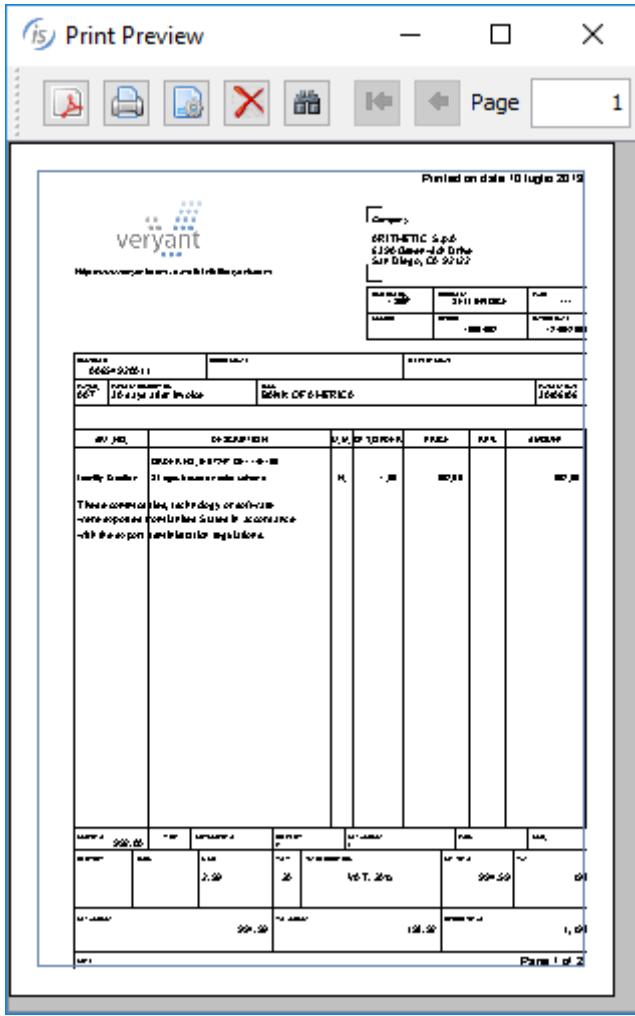


Push-button controls with the flat property set to true will now render all configured colors, no matter what LAF is active at runtime.

The printer preview dialog now allows you to set the color of the Printable Area Box. To read and write the color, two new methods class have been implemented in the com.iscobol.rts.print.SpoolPrinter class.

- `java.awt.Color getPrintableAreaBoxColor()`
- `setPrintableAreaBoxColor(java.awt.Color)`

The picture below shows the result of setting a different color for the preview printable area box.



## Hook on mouseover

Traditionally, there has been a single way to execute programs using a function key when a specific control has focus. This has usually been exploited to provide contextual help on specific controls, typically using the F1 key. The exception value to be used can be set with the following code:

```
SET EXCEPTION VALUE 1 TO ITEM-HELP
```

The following configuration variable is used to configure the program to run when the function key is pressed:

```
iscobol.help_program=myhelp
```

The program being run (in this example: myhelp), receives the information needed to determine which control has focus in the linkage section, as well as relative actions to be taken.

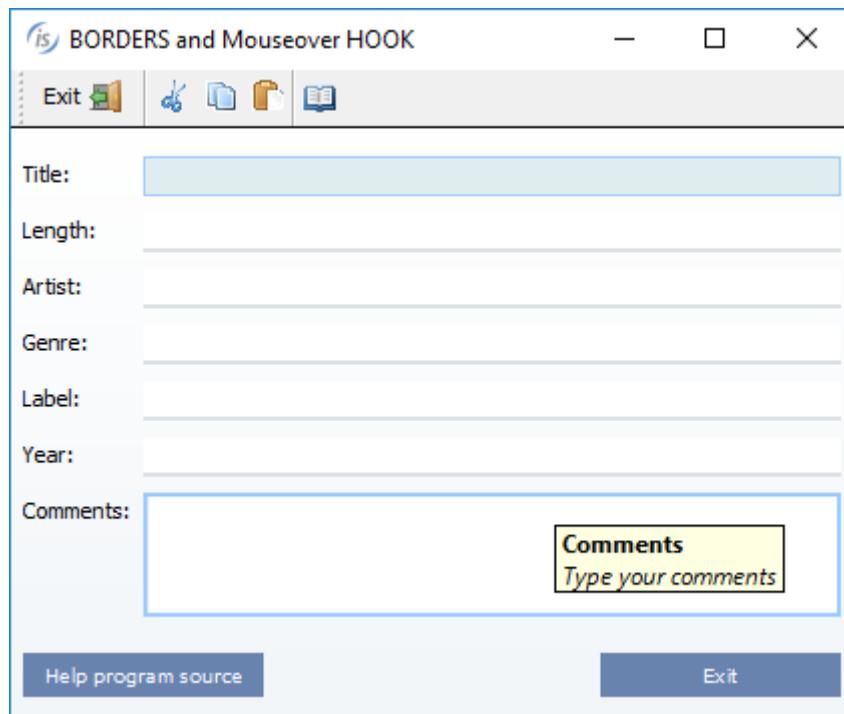
Now, the same functionality can be obtained by hovering the mouse on a control, without the user needing further interactions. A new property is available to set the delay from the moment the mouse stops and when the program is invoked.

```
iscobol.help_program_mouse_stop_delay=n
```

The called program has access, in the linkage section, to the following information:

- the control's handle
- the control's ID
- the control's help ID
- the handle of the control's owning window

The picture below shows the result of programmatically showing a hint when the configured program is run. Also shown is the flat push-button with background-color.



## Framework improvements

Performance of CALL statements and operations on c-treeRTG indexed files have been greatly improved. New configuration properties and new library routines have been introduced.

### Performance of CALL statements

Performance of CALL statements has been improved across the board, especially when using CALL/CANCEL (or C\$UNLOAD\_NATIVE for native libraries) statements, under all circumstances (CLASSPATH or code\_prefix). This is an additional improvement, adding to the optimizations already introduced in 2019 R1 edition, which was limited to the COBOL calls under code\_prefix.

The gain has been achieved by saving the information regarding the type of the executed CALL, whether it is a COBOL local call, a COBOL remote call, a library routine call (such as any of the C\$\*, CBL\_\*, P\$\*, WIN\$\* calls), or a Native Dynamic call or Native Static call for C functions. In a later call to the same function, the runtime already has access to its type, saving the time needed to determine it once again.

A table of performance gains is shown below. It shows a performance comparison between isCOBOL 2019R1 and isCOBOL 2019R2. The tests were run in Windows 10 64-bit on an Intel Core i5 Processor 4440+ clocked at 3.10 GHz with 8GB of RAM, using Oracle JDK1.8.0\_211. All times are in seconds.

The called COBOL program contains a linkage group data item with 50 child items. The number of CALL/CANCEL iterations is 10,000 for the COBOL program and 100,000 for Native calls and Library routines using a single parameter.

CALL statement type	2019 R1	2019 R2
CALL	0,01	0,01
CALL / CANCEL	1,60	0,04
Remote CALL	6,97	6,88
Remote CALL / CANCEL	16,96	10,22
CALL Client	6,38	6,28
CALL Client / CANCEL Client	12,21	9,68
CALL static C function	0,63	0,61
CALL dynamic C function	0,33	0,32
CALL / C\$UNLOAD native libraries	3,24	1,92
CALL library routines	0,03	0,03
CALL / CANCEL library routines	2,12	0,22
Total	50,48	36,21

### Performance of c-treeRTG indexed files

Performance of c-treeRTG indexed files has been improved constantly with a synergy between isCOBOL and the c-treeRTG file system. All COBOL statements that access indexed files (such as WRITE, REWRITE, DELETE, READ) have been improved. To achieve the maximum performance gain it is strongly recommended that you upgrade all your Veryant products, especially when older releases are still being used in production environments, as new features, performance improvements, and tweaks, are constantly being added to the entire suite.

A table of performance gains is shown below. It shows a performance comparison between different isCOBOL versions (2017R2, 2018R1 and isCOBOL 2019R2) with the embedded OEM c-treeRTG release available for each release (v.11.2, v11.5 and the latest v.11.6). The tests were run in the same environment and using the same hardware as the previous test. All times are in seconds.

The program used for the test is IO\_INDEXED.cbl, which is installed under sample\io-performances. The number of records used for this test is 500,000. The test is executed under the default c-treeRTG server configuration, and the property iscobel.file.index=ctreej is set in isCOBOL configuration to access c-tree files.

It's important to know that the c-treeRTG file system offers specific configuration settings both client-side and server-side that help in tuning the performances, depending on the architecture and needs, allowing further performance gains.

operation on ISCOBOL version:	2017 R2	2018 R1	2019 R2
Ctree version:	11.2.2.37148	11.5.2.51048	11.6.0.64778
WRITE	26,34	12,17	10,57
READ	3,77	3,29	3,14
REWRITE	11,90	8,03	7,36
DELETE	26,59	14,45	11,89
Total	68,60	37,94	32,96

## New configuration properties

Several new configuration properties have been implemented to enhance character based applications:

- *iscobol.terminal.cursor\_blink=n* to set the blinking cursor speed in character accepts, in milliseconds
- *iscobol.terminal.cursor\_color=n* to specify the cursor color number (0-15) in character accepts (default -1)
- new value in keystroke configuration: *edit=erase-all* used to clear all the fields in the character screen section in accepts

New configuration properties to enhance the graphical based application:

- *iscobol.gui.curr\_border\_color=n* to set the border-color of the currently focused entry-field
- *iscobol.gui.curr\_border\_width=n1 n2 n3 n4* to set the border-width of the currently focused entry-field
- *iscobol.gui.rollover\_border\_color=n* to set the border-color of entry-fields when the mouse hovers over them
- *iscobol.gui.rollover\_border\_width=n1 n2 n3 n4* to set the border-color of entry-fields when the mouse hovers over them
- *iscobol.gui.window\_title=xxx* to set the default window title when none is set in the COBOL source
- *iscobol.help\_program\_mouse\_stop\_delay=n* to set the time between when the mouse stops over a control and when the configured program is run, when using the "mouseover hook" feature. This can be useful, for example, to manage contextual help systems when existing COBOL applications need to be modernized without code changes.

Additional configuration properties:

- *iscobol.auto\_input\_mode=true* to activate IME (Input Method Editor) when the focus is on a field associated to a PIC N data-item. This works on both character and graphical user interface and it's useful when the application needs to input Chinese, Japanese, Korean and Indic characters
- *iscobol.key.accepted\_control\_characters=\u{hex value}* to specify control characters that should not be discarded during editing. This is useful for applications that use "barcode readers/scanners/guns" to insert values in the COBOL application. Multiple values can be specified by separating each with commas.

New ISUPDATER configuration properties:

- *swupdater.new\_jvm\_always=true* to always run the mainClass, defined in the swupdater.mainclass property, in a new JVM, even if there is no need to update any component on the machine
- *swupdater.jvm\_options=<java-options>* to specify the Java options for the newly started JVM after updating components, if the new\_jvm\_always property is set to false (or is not set), or always if the setting is set to true. If new\_jvm\_always is not set, the same options set on the invoking JVM are used on the newly instantiated one.
- *swupdater.items\_list.packagename=file1,file2...filen* and *swupdater.exclude\_items\_list.packagename=1|0* to exclude (when 1) or include (when 2) specific items in a package from the update process.

## isCOBOL Compiler

Starting from isCOBOL 2019R2, dynamic variables (X ANY LENGTH and OCCURS DYNAMIC) have been improved. New compiler configurations have been added to inject code for GUI controls and a new compatibility compiler option has been implemented.

### Dynamic variables

Dynamic variables allow developers to declare variables and arrays without knowing the size beforehand. One obvious advantage, on complex group level definitions, is to lower the memory footprint of the application. Typically, COBOL programmers have to choose a maximum value for an OCCURS array, which is most likely a random “big enough” value to handle each case. The maximum number of items assigned for the array may never be needed, but memory is assigned nonetheless.

Declaring the array as OCCURS DYNAMIC will allow dynamic sizing of the array, optimizing memory consumption.

The same can be applied to a PIC X(...) or PIC N(...) variable, where typically a maximum size needs to be determined. By using PIC X ANY LENGTH or PIC N ANY LENGTH, memory usage is optimized by allocating the right amount of memory needed to hold the contents.

Starting with isCOBOL 2019R2, group level variables containing dynamic items can be moved or compared to other compatible group level variables. Before this release, moving or comparing group level variables containing dynamic items were limited to non-dynamic items only.

For example, the following syntax defines 2 identical structures that contain dynamic child data items:

```
01 group1      group-dynamic.
  03 g1v1      pic x.
  03 g1v2      occurs dynamic capacity k-g1v2.
    05 g1v2a pic x.
    05 g1v2b pic 9.
  03 g1v3      pic x any length.
01 group2.
  03 g2v1      pic x.
  03 g2v2      occurs dynamic capacity k-g2v2.
    05 g2v2a pic x.
    05 g2v2b pic 9.
  03 g2v3      pic x any length.
```

The GROUP-DYNAMIC clause has been added to better describe how the structure is to be handled during moves and comparisons. This clause is not mandatory, and the compiler implicitly assumes it at compile time in groups that contain dynamic data items.

The advantage of this new approach is that now the following statements:

```
move group1 to group2  
if group1 = group2
```

will handle the dynamic items during moves and comparisons. This only works when the two structures are mirrors of each other. This eases the code rewrite needed when moving code from static structures to dynamic ones.

Other improvements allow developers to retrieve the current capacity of an occurs dynamic variable, without the need to declare the capacity in the data division, as the code below shows.

```
set curr-size to capacity of g2v2
```

This is very useful when the programs need to retrieve different capacities in nested occurs dynamic items.

The SORT statement, which can be used to sort data in an array structure, now supports sorting data structures containing dynamic occurs data items, as shown below:

```
sort cust-array on descending key cust-name  
          on ascending key cust-city
```

ANY LENGTH data items can now be pre-allocated using the WITH SIZE clause on the INITIALIZE statement. This will initialize the variable with the amount of spaces defined in the with size clause.

For example

```
initialize var-1
```

initializes var-1 with a zero-length size, while

```
initialize var-2 with size 3
```

initializes var-2 with 3 spaces.

During the migration from static occurs to dynamic occurs inside groups, the compiler now issues a warning to identify code that is potentially affected by the changes in the handling of dynamic variables. For example, the following lines of code:

```
display group1  
if group1 = "ab1c"
```

will result in a warning being issued if group1 contains dynamic variables, to warn the developer that the statements could fail. The warning issued is as follows:

--W: #257 Dynamic items will be ignored: GROUP1

## Compiler code injection

New compiler configuration settings are implemented to inject COBOL code in all controls of a specific type at compile time.

```
iscobol.compiler.gui.<control-name>.defaults=...
```

where <control-name> can be any of the following: bar bitmap, check\_box, combo\_box, date\_entry, entry\_field, frame, grid, java\_beans, label, list\_box, push\_button, radio\_button, ribbon, scroll\_bar, slider, status\_bar, tab\_control, tree\_view, web\_browser, window, tool\_bar.

This feature simplifies the modernization process for GUI applications and reduces developing efforts.

As an example, when compiling the following screen section controls:

```
01  s1.  
 03 ef1 entry-field  
      line 2 col  2 size 10.  
 03 ef2 entry-field  
      line 2 col 14 size 10.  
 03 ef3 entry-field  
      line 2 col 26 size 10.  
 03 pb1 push-button  
      line 5 col 10 size 10  
      title "Save" exception-value 1.
```

with the following compiler configuration:

```
iscobol.compiler.gui.push_button.defaults=flat, background-color -14675438  
iscobol.compiler.gui.entry_field.defaults=border-color rgb x#dae1e5, \  
                                border-width (0 0 2 0 )
```

the compiler will compile the source code as if it were written as:

```
01  s1.  
 03 ef1 entry-field  
      border-color rgb x#dae1e5,  
      border-width (0 0 2 0 )  
      line 2 col  2 size 10.  
 03 ef2 entry-field  
      border-color rgb x#dae1e5,  
      border-width (0 0 2 0 )  
      line 2 col 14 size 10.  
 03 ef3 entry-field  
      border-color rgb x#dae1e5,  
      border-width (0 0 2 0 )  
      line 2 col 26 size 10.  
 03 pb1 push-button  
      flat, background-color -14675438  
      line 5 col 10 size 10  
      title "Save" exception-value 1.
```

Code injection also affects controls created with single display statement

```
display push-button line 5 col 25 size 10
      title "End" exception-value 27
      handle in h-pb.
```

With code injection, an entire application can be recompiled without code changes, and changing the configuration variables can result in a completely different looking application, allowing modernization to take place without altering the source code.

Code injection works by inserting the text in the configuration variables in the source code where controls are declared or created. Syntax errors in the configuration variables will result in compilation errors.

### Compatibility enhancements

A new compiler option, –cr, has been added to support new syntax, enhancing compatibility with RM/COBOL v8 or greater. This option allows us to support the following additional syntax:

- PROGRAM-ID and WHEN-COMPILED special registers
- ACCEPT data-item FROM DATE-COMPILED
- specific management for ERASE clause and DISPLAY without LINE for RM/COBOL compatibility

New library routines are now supported to simplify the migration from RM/COBOL:

- C\$MBAR to display a menu bar on the active window
- C\$RBMENU to create a pop-up menu on the active window
- C\$SBAR to display a single panel status-bar
- C\$SCRD to read character text
- C\$SCWR to write character text
- C\$TBAR to display a tool-bar on the active window
- WOWGETWINDOWTYPE to inquire for the window type
- C\$SHOW to hide and show the main window

## isCOBOL Remote Debugger

Starting from isCOBOL 2019R2, the isCOBOL Remote Debugger allows stopping execution only on breakpoints specifically set, instead of automatically breaking on the first statement of the first program compiled in debug mode. This is very useful in multithreaded environments, such as a J2EE container. To activate this feature, a new configuration setting needs to be set on the server side where the COBOL program is running:

```
iscobol.rundebug.auto_pause=false
```

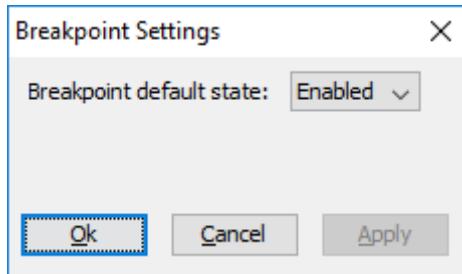
this setting works in conjunction with the existing:

```
iscobol.rundebug=1
```

With the new configuration active, when Remote Debugger starts up and connects to the isCOBOL program's process, the execution can be interrupted or breakpoints can be set using debugger commands such as pause, break, b0 or m0.

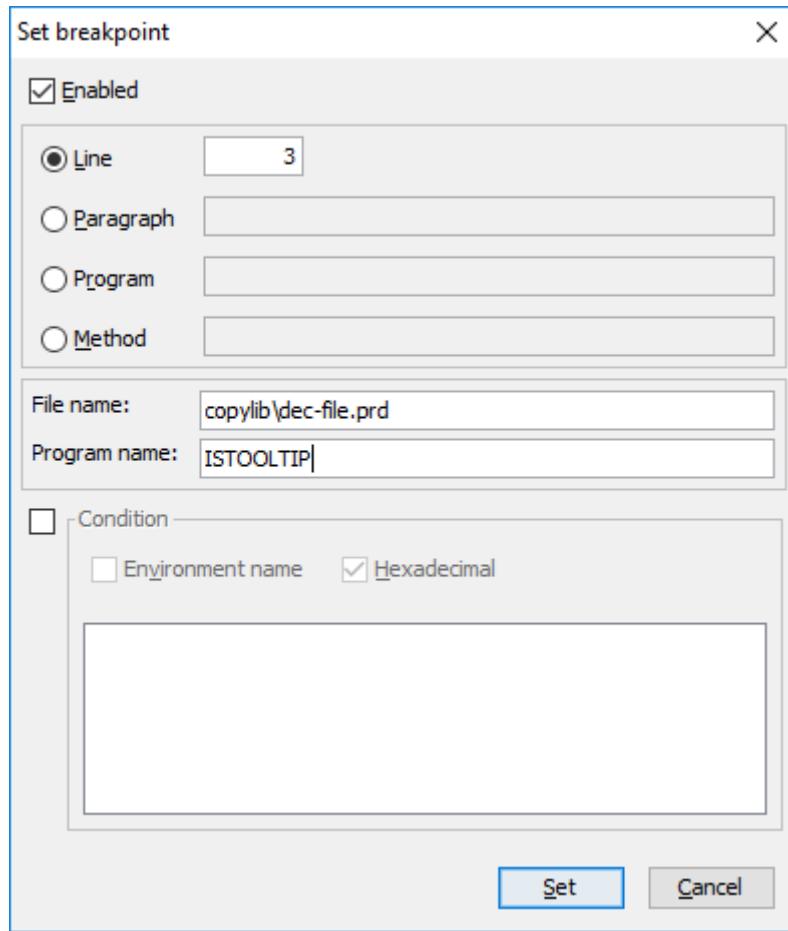
The rundebug.auto\_pause configuration setting is set to true by default, to maintain the existing behavior.

A new dialog accessible from the "settings" menu has been designed to allow setting the default state of new breakpoints, using a combo-box, as shown the picture below. This provides an easy way to set the initial state of newly added breakpoints.



The "Set breakpoints" dialog, depicted in the picture below, has been improved by allowing you to set a breakpoint for a specific program on a specific file name, typically a copy file. Such a breakpoint will trigger on that file name, at the specified line number, only for a specific program.

If a program name is not specified, the breakpoint will trigger at any program that contains the specific copy file.



## isCOBOL EIS

isCOBOL EIS, Veryant's solution to write web-enabled COBOL programs, is constantly updated to provide more comprehensive web solutions. In isCOBOL 2019R2 webClient has received many updates, the HttpClient class can consume existing web services with attachments, and XMLStream has more powerful xml file handling.

### webClient enhancements

webClient will now alert the user when an admin or a support user is viewing or recording their session. webClient's auditing capabilities now log support activities, showing the user name that started viewing, recording, or taking control of the session.

A new option has been added to the application configuration page that allows automatic recording of sessions that are being viewed by authorized support and admin users. With this setting enabled, whenever a user starts viewing a session, it will automatically be recorded.

Session recording can be stopped and resumed multiple times within a support session, allowing finer control of what is being recorded.

## HTTPClient improvements

The `HTTPClient` class is very useful to allow COBOL programs to interact with Web Services. It has been updated to handle Multipart responses by providing several new methods to get the list of received attachments, the list of attribute names for a specific attachment, the values for a specific attribute of a specific attachment, and the content of a specific attachment.

The new methods are shown below:

- `public void getResponseAttachmentIDs(destination)`
- `public void getResponseAttachmentAttrNames(id, destination)`
- `public void getResponseAttachmentAttr(id, attrName, destination)`
- `public void getResponseAttachmentBody(id, destination)`

## XML with qualified tag names

New methods have been implemented in the `XMLStream` class to better handle XML files when namespaces are involved, making it easier to exchange XML files with third parties.

All the existing methods that write XML files now support an optional Boolean parameter that, when set to true, will generate qualified names in the resulting XML.

This following is the list of methods that support the additional `writeQualifiedTagNames` parameter:

- `public void write ( Xml-Destination, writeQualifiedTagNames)`
- `public void writeToFile ( Xml-Destination, writeQualifiedTagNames)`
- `public void writeToPrintWriter ( Xml-Destination, writeQualifiedTagNames)`
- `public void writeToStream ( Xml-Destination, writeQualifiedTagNames)`
- `public void writeToStringBuffer (Xml-Destination, writeQualifiedTagNames)`

The sample code below shows the difference in generated XML file when using the new method with the `writeQualifiedTagNames` parameter.

Below is the copy file generated by running `stream2wrk` utility with the command:

```
stream2wrk xml book.xml -p book-
```

```

>>SOURCE FORMAT FREE
*> XML File: book.xml
*>
*> Generated by isCOBOL
*> Stream2Wrk options: -p book-

01 book-books identified by 'books'
    namespace 'http://somebooksite.com/book_spec'.
03 book-book identified by 'book'
    namespace 'http://somebooksite.com/book_spec'
    occurs dynamic capacity book-book-count.
05 book-title identified by 'title'
    namespace 'http://somebooksite.com/book_spec'.
07 book-title-data pic x any length.
05 book-author identified by 'author'
    namespace 'http://somebooksite.com/book_spec'.
07 book-author-data pic x any length.
05 book-email identified by 'email'
    namespace 'http://somepublishingsite.com/spec'.
07 book-email-data pic x any length.
>>SOURCE FORMAT PREVIOUS

```

When invoking the write method without specifying the writeQualifiedTagNames parameter:

```
obj-xml:>write("output1.xml")
```

or using the equivalent new method with writeQualifiedTagNames set to false:

```
obj-xml:>write("output3.xml", false)
```

The XML file created will not contain qualified tag names, as shown below.

```

<?xml version="1.0" encoding="UTF-8"?>
<books xmlns="http://somebooksite.com/book_spec">
    <book>
        <title>The Dream Saga</title>
        <author>Matthew Mason</author>
        <email xmlns="http://somepublishingsite.com/spec">
            author@sidharta.com.au
        </email>
    </book>
    <book>
        <title>Sherlock Holmes - I</title>
        <author>Arthur Conan Doyle</author>
        <email xmlns="http://somepublishingsite.com/spec">
            author@GeorgeNewnes.com
        </email>
    </book>
</books>

```

Invoking the write method with the writeQualifiedTagNames parameter set to true:

```
obj-xml:>write("output2.xml", true)
```

will generate an XML file with qualified tag names, as shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:books xmlns:ns1="http://somebooksite.com/book_spec"
            xmlns:ns2="http://somepublishingsite.com/spec">
    <ns1:book>
        <ns1:title>The Dream Saga</ns1:title>
        <ns1:author>Matthew Mason</ns1:author>
        <ns2:email>author@sidharta.com.au</ns2:email>
    </ns1:book>
    <ns1:book>
        <ns1:title>Sherlock Holmes - I</ns1:title>
        <ns1:author>Arthur Conan Doyle</ns1:author>
        <ns2:email>author@GeorgeNewnes.com</ns2:email>
    </ns1:book>
</ns1:books>
```

## isCOBOL 2019 Release 1 Overview

### Introduction

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

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

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

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

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

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

Details on these enhancements and updates are included below.

### Support for Java 11 and OpenJDK

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

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

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

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

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

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

- *swupdater.net.ssl.trust\_store*=keystore to set the keystore
- *swupdater.net.ssl.trust\_store\_password*=password to set the password of the keystore
- *swupdater.http.ignore\_certificates=true* (default is false) to ignore invalid certificates

## Media Type application

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

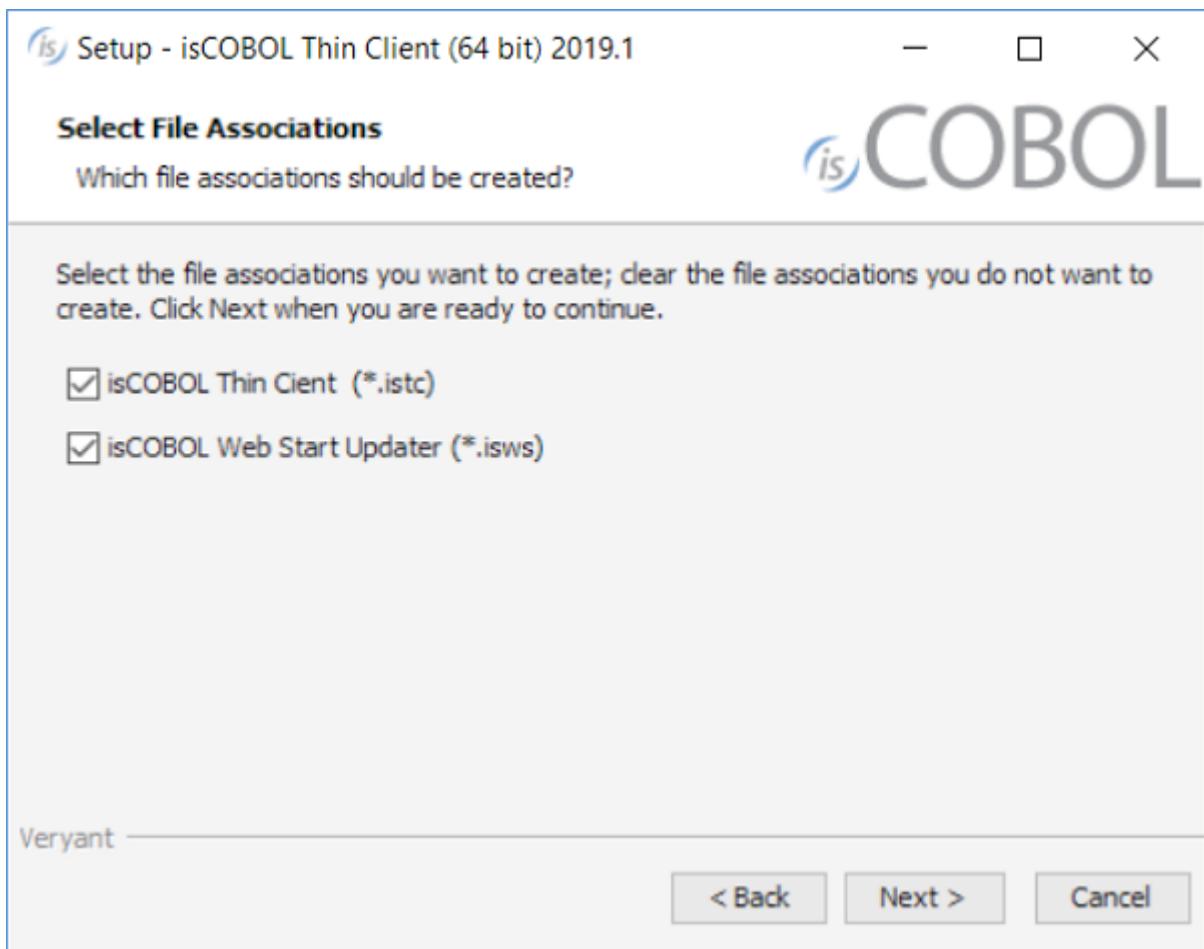
These file suffix registrations, done from the isCOBOL installation setup screen, will simplify the execution and the automatic update for the end user. The registered files contain all the isCOBOL properties needed to set up the desired execution of isCOBOL THIN and isUPDATER technologies on the client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

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

The new file extension associations are:

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

The new isCOBOL 2019R1 installation screen prompts you to define what mime types need to be registered on the operating system.



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

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

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

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

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

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

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

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

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

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

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

## Responsive Layout Manager

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

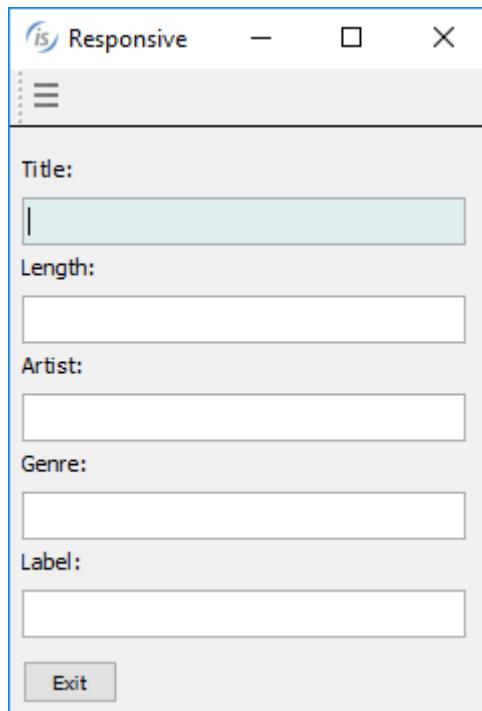
- PB-EXIT is hidden for the “small” breakpoint and visible on all other breakpoints
- PB-MENU is visible for the “small” breakpoint and hidden for all other breakpoints
- NO-SCALE clause replaces the default LM-SCALE behavior MOVE-BOTH-ANY

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

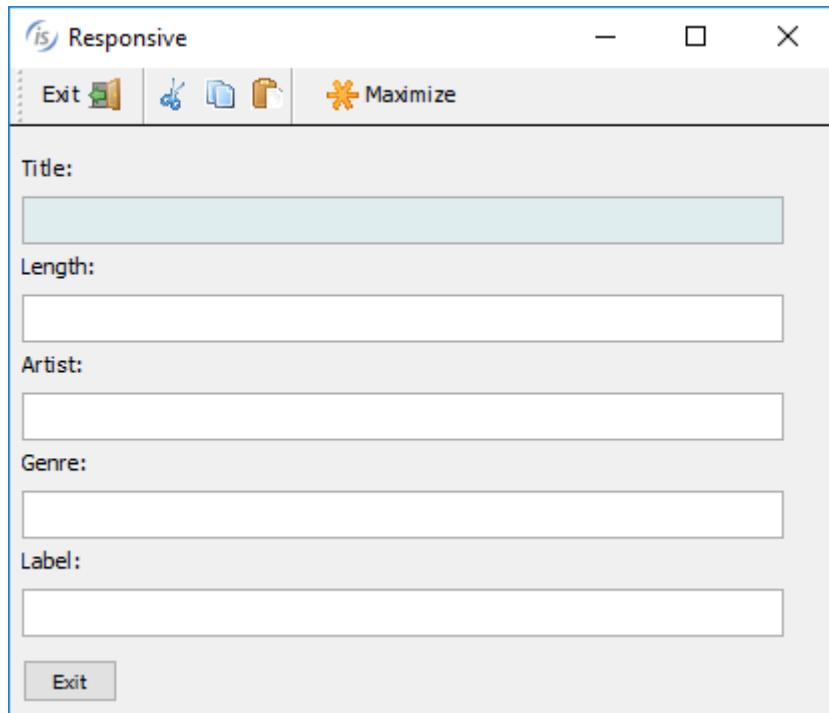
To enable responsive layout, the DISPLAY WINDOW and DISPLAY TOOL-BAR statements also need to use the LAYOUT-MANAGER property:

```
display standard graphical window resizable
  layout-manager responsive-layout
display tool-bar moveable
  layout-manager responsive-layout
```

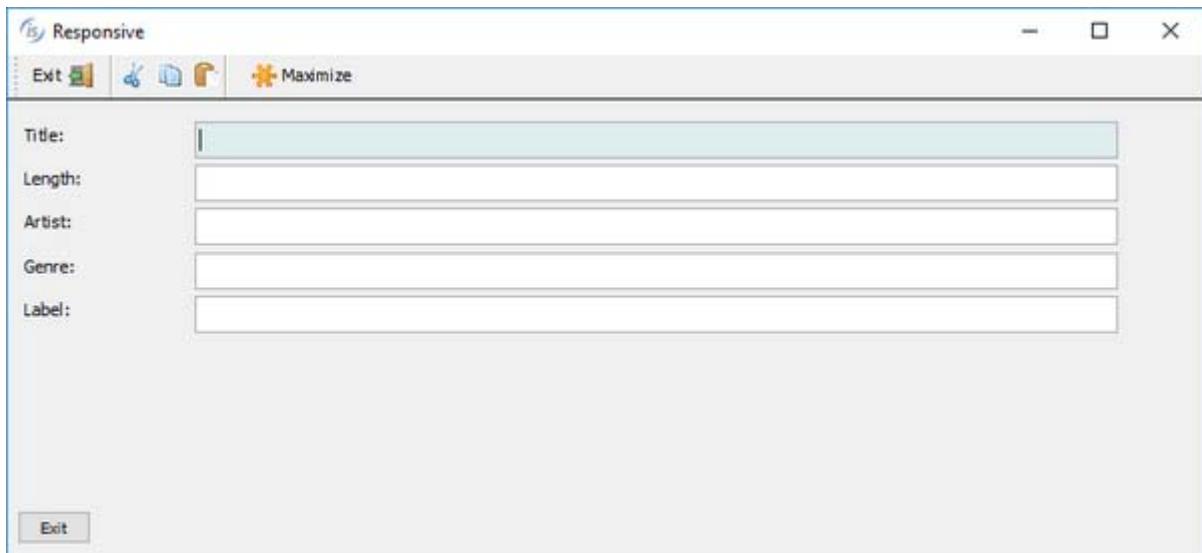
The picture below shows how the program runs under small devices according to all “small” breakpoint rules defined in LAYOUT-DATA property with the typical “hamburger” menu.



The picture below shows how the program runs under medium devices. Now the toolbar contains new buttons (visible on “medium” breakpoint) and the “hamburger” button menu is hidden.



The picture below shows the same program running on a large device, with space enough to have a label and an entry-field on the same line.

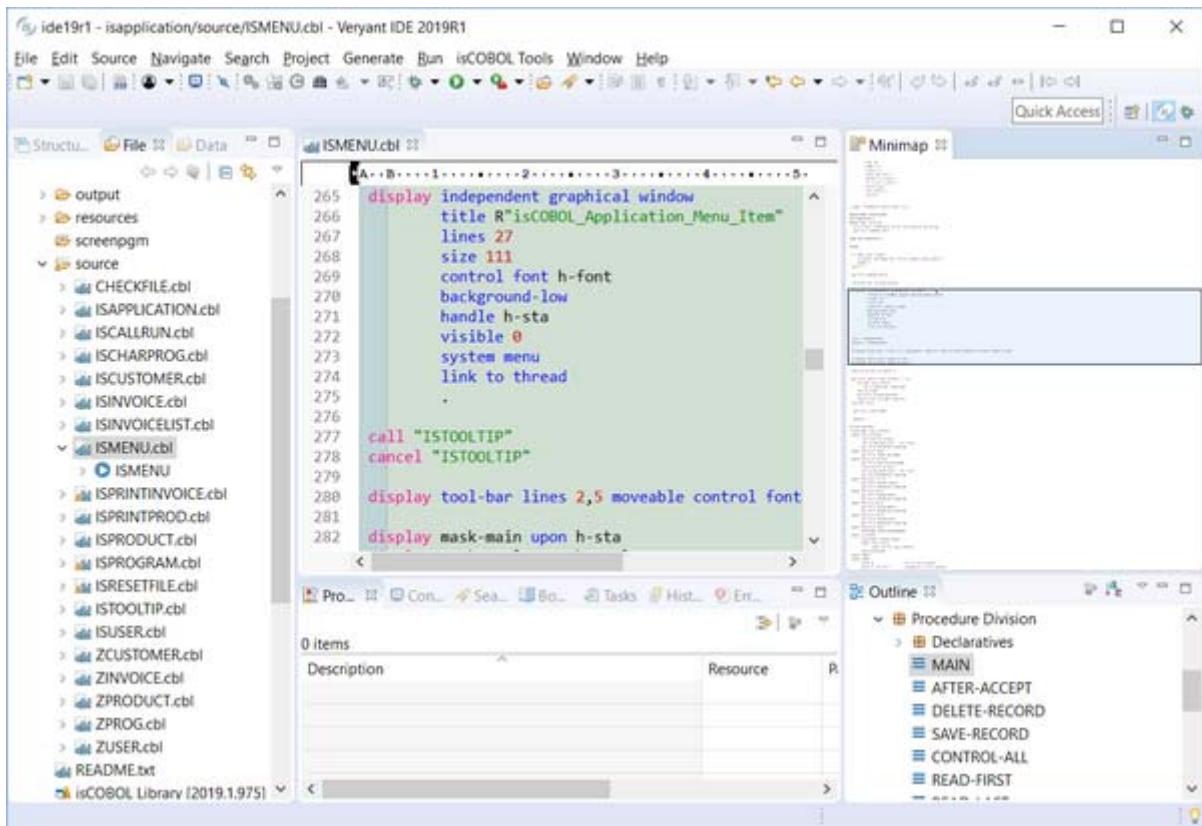


## isCOBOL IDE Enhancements

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

## Eclipse features

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



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

## COBOL-WOW support

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

All of COBOL-WOW GUI widgets are supported and written in Java for 100% portability across environments, with an updated and modern look. isCOBOL IDE support for COBOL-WOW provides the following UI widget: Command Button, Check Box, Option Button, Static Text, List Box, Combo Box, Vertical, Scroll, Horizontal Scroll, Toolbar, Timer, Month Calendar, Rounded Rectangle, Eclipse Edit Box, Group Box, Bitmap, Animation Control, Progress Bar, Track Bar, Status Bar, Up/Down Control, Tab Control, Data Time Picker, Line, Rectangle.

Also, most of COBOL-WOW routines are provided in order to have a seamless execution of generated COBOL source code: Axbindeventarguments, Axdomethod, Axunbindeventarguments, Checkmenuitem, Closewindow, Deletemenu, Drawmenubar, Enablemenuitem Enablewindow, Findwindow, Getactivewindow, Getcursorspos, Getenvironmentvariable, Getfocus, Getmenu, Getsubmenu, Getwindowsdirectory, Ischild, Iswindow, Messagebeep, Messagebox, Modifymenu, Openicon, Sendmessage, Setactivewindow, Setfocus,

Showwindow, Winexec, Wowadditem, Wowclear, Wowclearwaitcursor  
 Wowcreatewindow, Wowdestroywindow, Wowdiscardevents, Wowgetfocus, Wowgetindexprop,  
 Wowgetmessage, Wowgetnum, Wowgetprop, Wowinitallcontrols, Wowinitcontrol, Wowmessagebox,  
 Wowmove, Wowmulticontrolgetprop, Wowmulticontrolsetprop, Wowpeekmessag, Wowrefresh,  
 Wowremoveitem, Wowresetwaitcursor, Wowsetfocus, Wowsetindexprop, Wowsetnextctrl, Wowsetnum,  
 Wowsetprevctrl, Wowsetprop, Wowsetstriptrailing, Wowsetwaitcursor, Wowversion1.

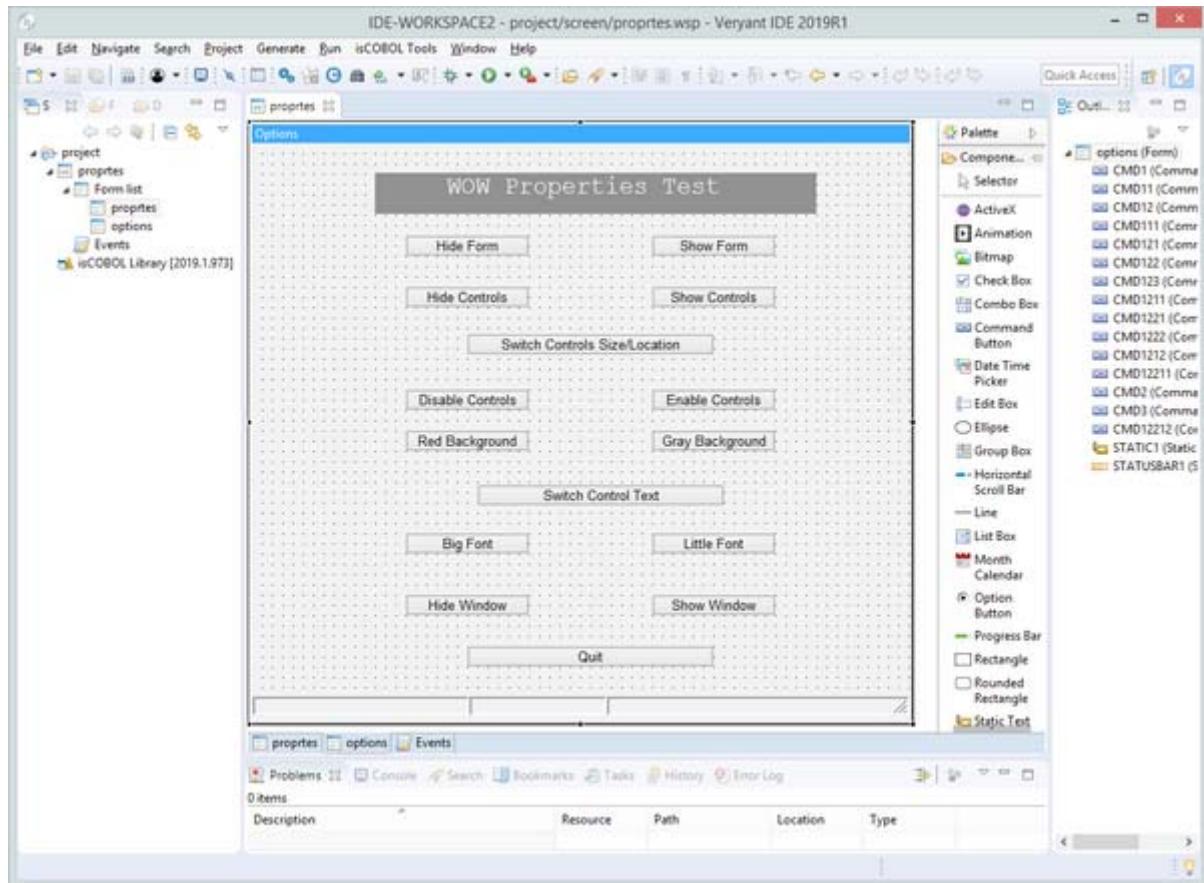
Once your code is in the isCOBOL Evolve environment the possibilities are endless, from object-oriented COBOL programming to all the benefits of working within the Java ecosystem. This includes the close-to-infinite number of libraries and toolkits you can use to tackle every conceivable task as well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

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

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

The picture below shows a COBOL WOW program imported in isCOBOL IDE.

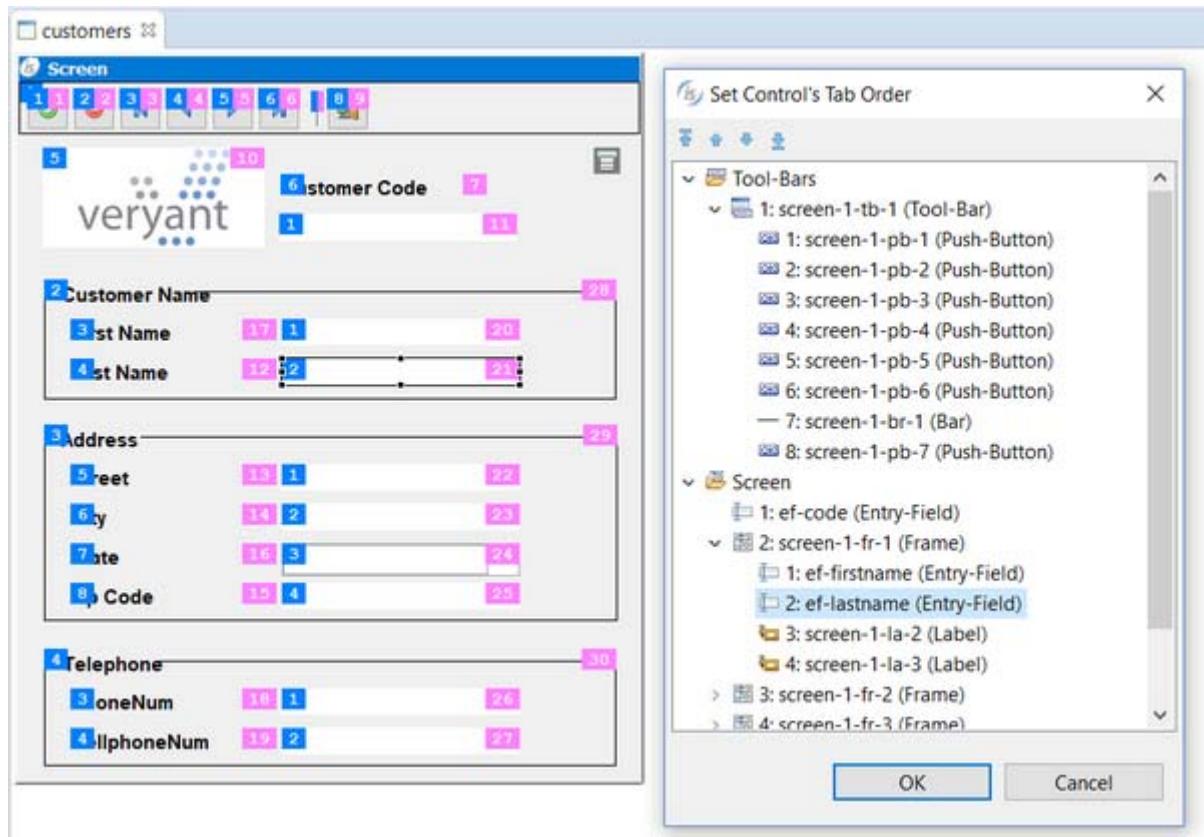
You can see the WYSWYG GUI Painter, the list of supported GUI widgets, the event editor and more.



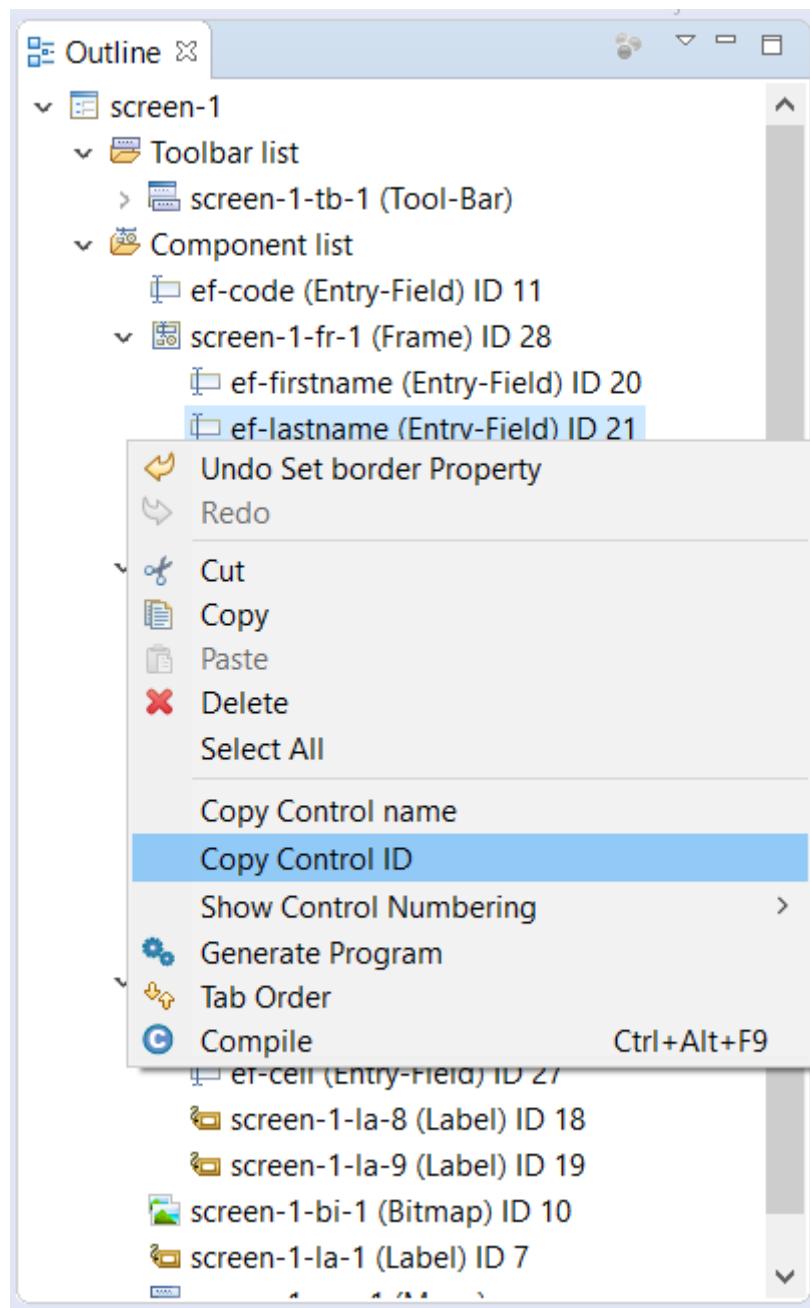
## New screen painter features

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

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



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



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

# Framework improvements

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

## Performance of CALL statements

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

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

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

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

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

The code\_prefix tests under 2019R1 are in conjunction with `code_prefix.reload=false`

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

## New configuration properties

New configuration properties have been introduced in the Framework:

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

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

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

- *iscobol.default\_program*=PGM to specify the main program to execute when it's not passed
- *iscobol.default\_options*=options to specify the options for isCOBOL execution (standalone or thin client)

Additional configuration properties:

- *iscobol.call\_run.sync*=true to execute the CALL RUN synchronously instead of asynchronously
- *iscobol.esql.indicator\_trunc\_on\_call*=false to set the indicator variable to 0 when the stored procedure's output parameter value doesn't fit the host variable
- *iscobol.file.indd*=*myindd* to associate custom file handlers to the files specified by INDD directive
- *iscobol.file.outdd*=*myoutdd* to associate custom file handlers to the files specified by OUTDD directive
- *iscobol.gui.entryfield.notify\_change\_delay*=n to affect all entry-fields
- *iscobol.key.default\_shortcuts\_enabled*=true to intercept shortcuts like Ctrl+C
- *iscobol.upper\_lower\_method*=n default value 1, where n can be:
  - o 1, to use the method of String.toUpperCase / toLowerCase
  - o 2, to use the method of Character.toUpperCase / toLowerCase

## Enhanced push button title position

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

The picture below shows the result of the following code snippet which applies the new position to the third button:

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



## New library routines

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

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

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

server side.:

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

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

- CBL\_EXEC\_RUN\_UNIT can be used to run a unit that inherits the environment variables of the calling program
- C\$ENCRYPT and C\$DECRYPT can encrypt and decrypt a file using several cryptographic algorithms, as configured in the *iscobol.crypt.algorithm* property, such as:

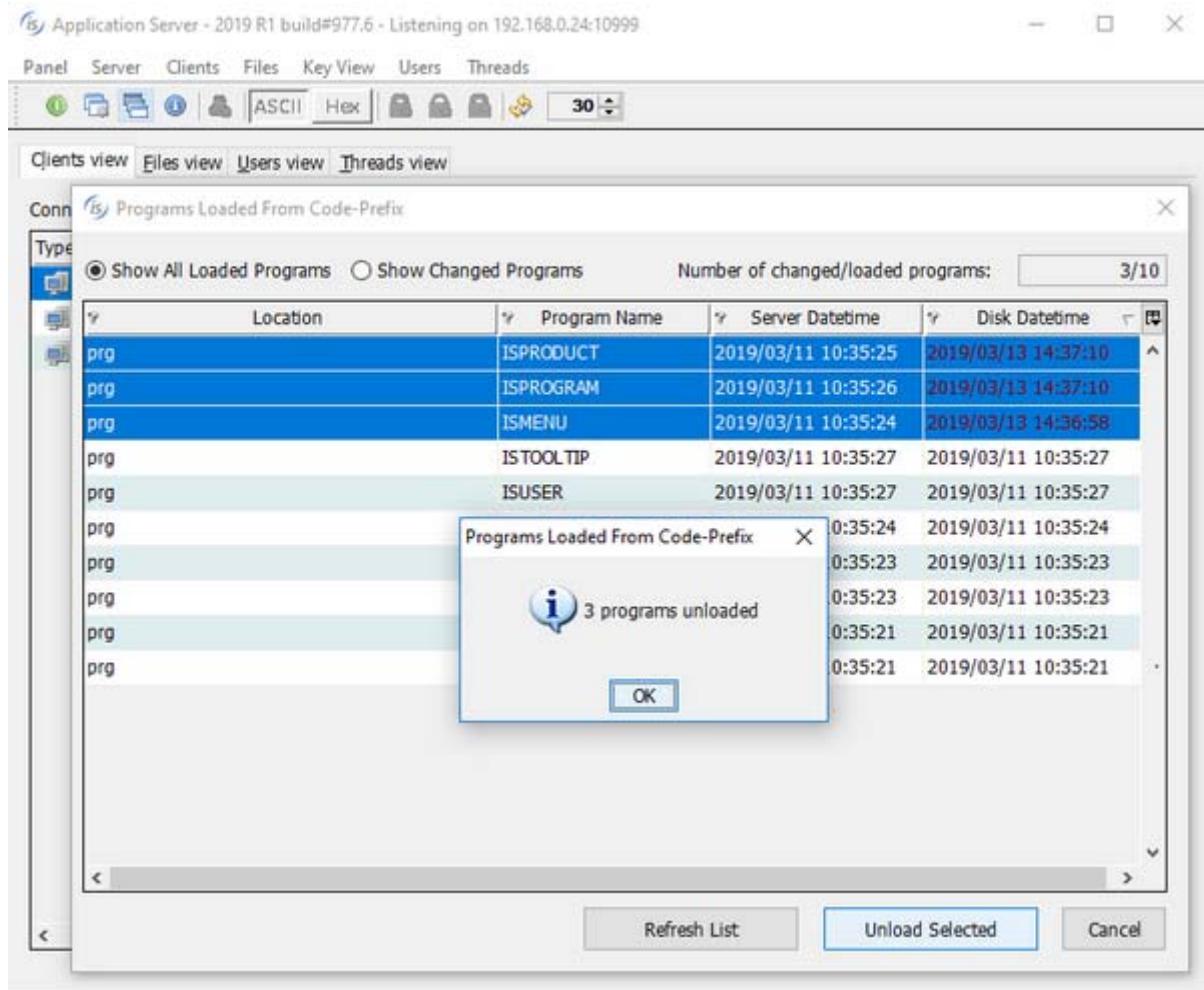
AES	Advanced Encryption Standard as specified by NIST in FIPS 197
AESWrap	The AES key wrapping algorithm as described in RFC 3394
ARCFOUR	A stream cipher believed to be fully interoperable with the RC4 cipher
Blowfish	The Blowfish block cipher designed by Bruce Schneier
CCM	Counter/CBC Mode, as defined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

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

## isCOBOL Server

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

To take advantage of the unload feature, you need to configure the application to be executed with "code\_prefix" class loader having *iscobol.code\_reload* set to false.



## isCOBOL Compiler

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

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

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

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

## New OOP Syntax

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

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

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

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

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

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

## Auto boxing on primitive types

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

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

## New Compiler options

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

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

will be translated as:

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

Other compiler options added in isCOBOL 2019 R1:

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

## New Syntax for compatibility

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

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

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

The code below is now supported:

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

## isCOBOL EIS

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

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

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

### Custom code

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

```
*>start {iscobol}http-to-linkage  
move PAR1-in to intPAR1;;  
*>end {iscobol}http-to-linkage  
*> This is my custom code written outside the Tagged Areas  
CALL "MYPROG" USING intPAR1  
*>start {iscobol}call  
call "ws-info" using intPAR1  
*>end {iscobol}call
```

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

### Boolean data type

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

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

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

The JSON generated:

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

## isCOBOL Database Bridge

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

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

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

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

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

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

```
iscobol.fileSEQUENTIAL=easydb  
iscobol.fileLINESEQUENTIAL=easydb  
iscobol.fileRELATIVE=easydb
```

# isCOBOL 2018 Release 2 Overview

## Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2018 R2.

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

isCOBOL 2018 R2 includes several enhancements to GUI controls, such as automatic search and filter in the Grid control, and other new features and compatibility options.

Developer productivity has been enhanced by providing improved code navigation while editing in the IDE and debugging programs, and new debugger features have been implemented.

isCOBOL EIS has been improved to allow tighter integration with external programs.

Support for Asian markets has been improved as well, allowing easy migration to isCOBOL.

Details on these enhancements and updates are included below.

## isCOBOL IDE Enhancements

The isCOBOL 2018 R2 IDE improves the editor by displaying useful additional info to COBOL developers and improving the usability of existing features.

### Editor features

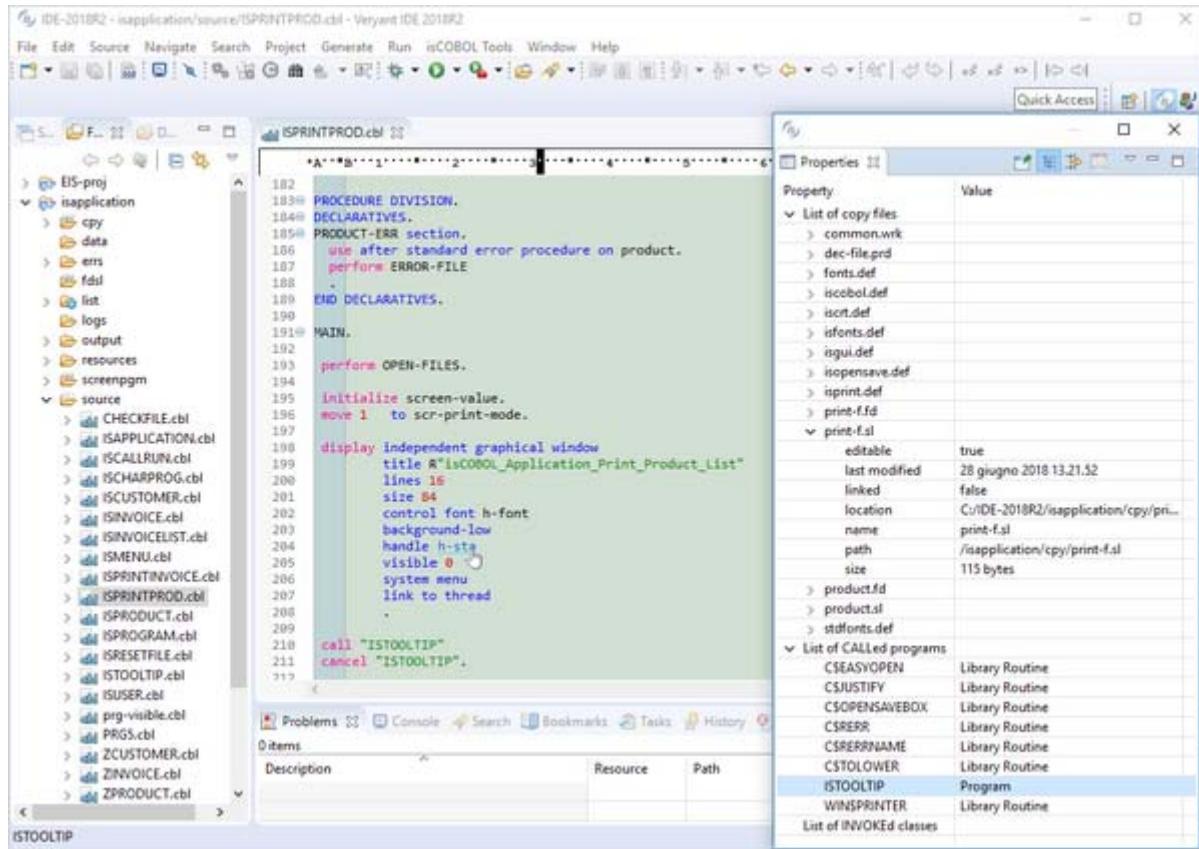
Opening a source file will now list important information in the “Properties” window, by default located on the bottom right of the IDE window, such as:

- a list of copy files used
- a list of called programs
- a list of invoked classes

By double-clicking on a line in the property window, the selected copy file or program will be opened in the Editor if the source is available in the current workspace. For example, double-clicking the selected line shown in Figure 1, *IDE Property window and hyperlink*. will open the “ISTOOLTIP.cbl” source file.

The feature that creates a “Hyperlink” on a variable, paragraph or screen-control name in the editor has been simplified. By clicking on the name when the mouse has the “hand” shaped pointer, the IDE will position the editor on the declaration, whether it is located in the same source file, or in a different copy file.

**Figure 1.** IDE Property window and hyperlink

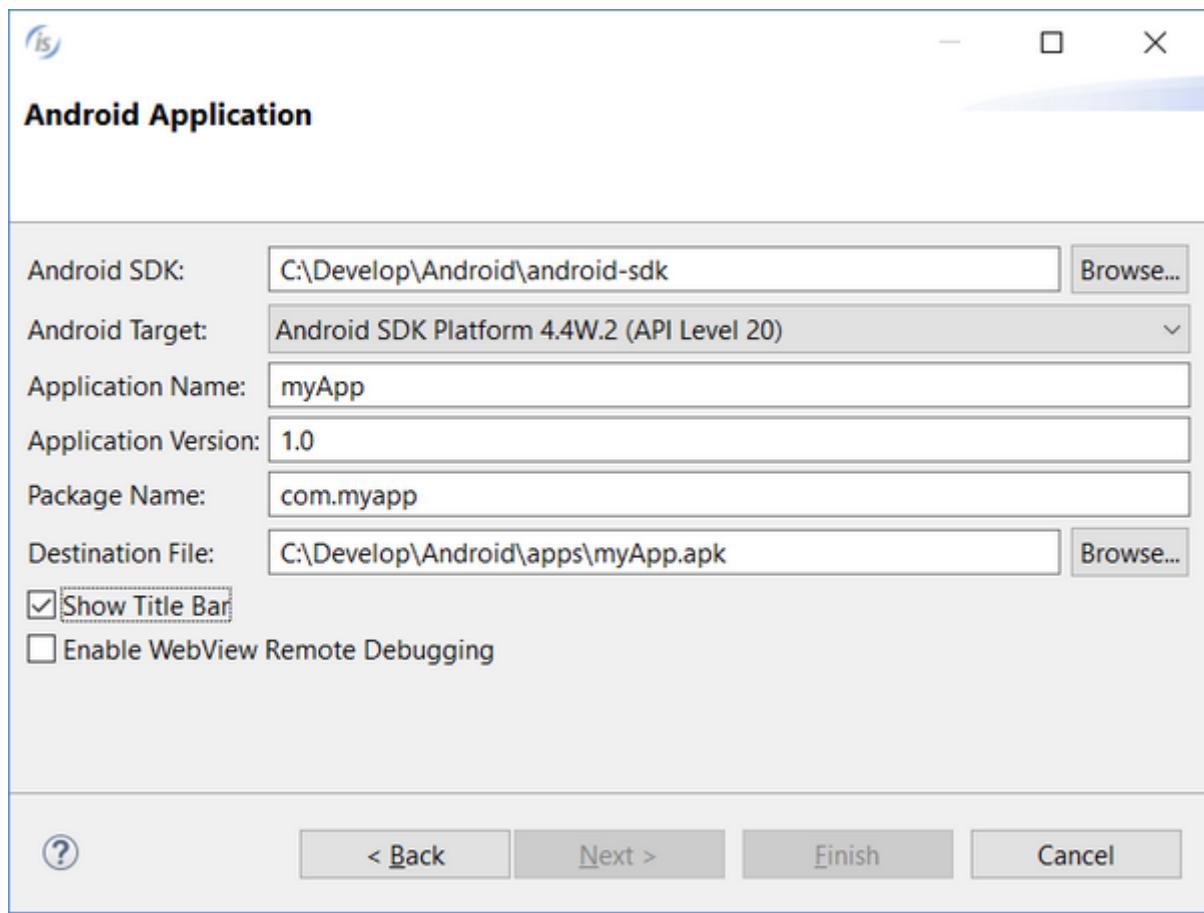


A program can now be compiled even when the current editor file is a copy file, provided the copy file was opened from its parent source file, while previous versions required a program source file to be active.

## Export to Android

The isCOBOL IDE now allows you to customize the application title bar, when exporting the project as an Android application. If you check the “Show Title bar” option as shown in Figure 2, *Android title bar*, then an Application Bar will be rendered at the top of the HTML UI of the exported application, displaying the application name.

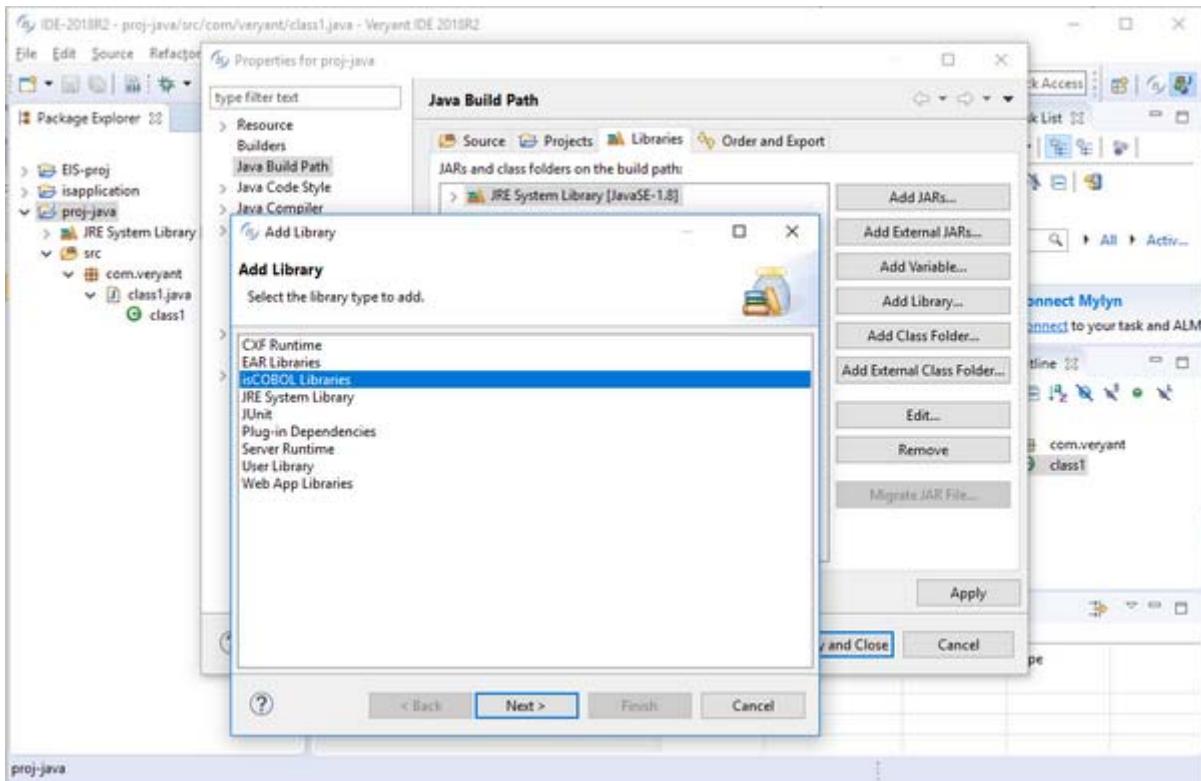
**Figure 2.** Android title bar



### Simpler Java integration

The “isCOBOL libraries” item has been added to the “Add Library” button of the “Java Build Path” section in a Java Properties page, allowing simple integration of isCOBOL libraries to a Java application, as shown in Figure 3, *Add isCOBOL Libraries*. This replaces the need to manually add the isCOBOL libraries to the Java project’s ClassPath, required in previous versions.

**Figure 3.** Add isCOBOL Libraries



## New User Interface Features

The GRID control has been enhanced with automatic search and filter features. Screen sections are now dynamic, allowing controls to be added and removed at runtime.

Other minor enhancements are designed to improve User Interface handling.

### GRID enhancements

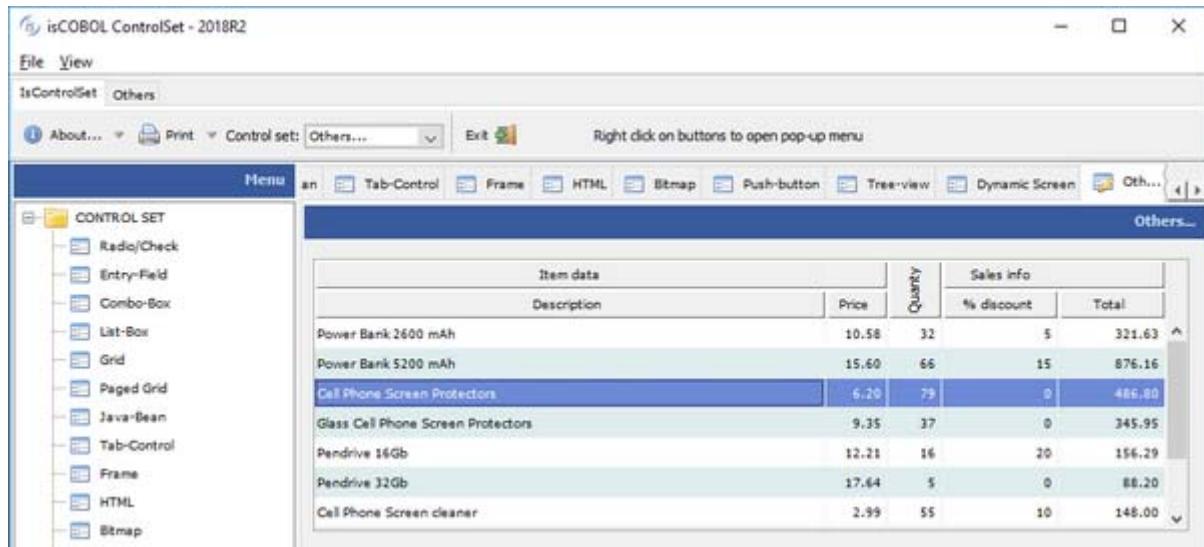
Grid searching is enabled by default and can be activated by the user at runtime by pressing Ctrl-F. A NO-SEARCH style is available to disable the feature when needed.

When the search function is activated, a panel is shown on the top portion of the grid to let the user search the grid for data. In the panel the user can enter the search text or select a previously used one. When the Find button is pressed, the occurrences of the search text are highlighted inside the grid. The Clear button allows the user to reset the search text, and the 'X' button allows the user to close the search panel.

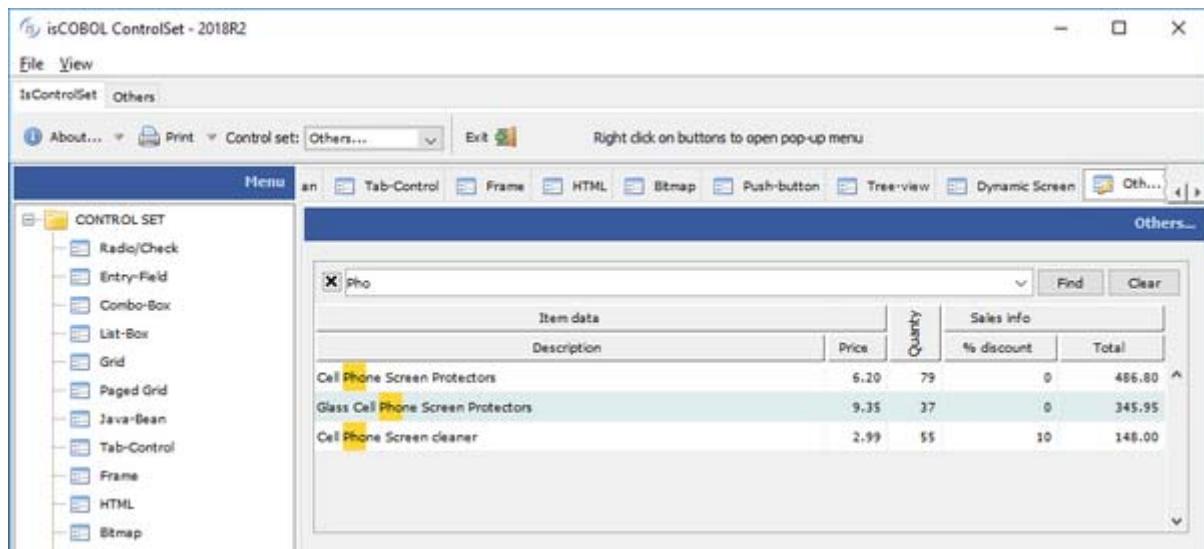
The HEADING-MENU-POPUP property of grid now also supports the values GRHM-FIND-ON-RIGHT-CLICK and GRHM-FIND-ON-BUTTON to add the new Find item on the automatic heading menus.

Figure 4, *Grid before activating the search function*, shows the grid while the user is browsing, while Figure 5, *Search on grid* shows the search panel displayed when the user presses Ctrl-F while the grid has focus. This allows the user to search all occurrences of the given text.

**Figure 4.** Grid before activating the search function



**Figure 5.** Grid with the search function active



A new style named FILTERABLE-COLUMNS is supported on grids, to allow data filtering based on column content, as shown in Figure 6, Grid filtering. When activated on a column, a popup window will show every distinct occurrence of the column values, allowing the user to choose one or more values to use for filtering rows. Filters can be added to multiple columns and removed as needed. An icon is displayed on the column header to show an active filter.

```
05 h-grid, grid
filterable-columns
...
```

**Figure 6.** Grid filtering

The screenshot shows a software interface for filtering a grid of data. On the left, there is a sidebar with a list of items, each preceded by a checkbox. Some items have a blue highlight. The main grid on the right displays rows of data with columns labeled Description, QT, Price, %, Date, and YY/n. The first few rows of the grid also have blue highlights corresponding to the selected items in the sidebar. The bottom of the screen shows a footer with the date '7/2018@15:39' and a double-click instruction.

	Description	QT	Price	%	Date	YY/n
01	All	16	223.58	60	01/01/2013	<input type="checkbox"/>
02	COREL CorelDraw Graphic Suite 11 (Full)	05	238.80	100	01/02/2013	<input checked="" type="checkbox"/>
03	<b>COREL CorelDraw Graphic Suite 12 (Full)</b>	52	123.96	90	01/03/2012	<input type="checkbox"/>
04		23	99.00	40	01/04/2013	<input checked="" type="checkbox"/>
05	COREL CorelDraw Graphic Suite 12 (Upgrade)	50	50.00	60	01/05/2013	<input type="checkbox"/>
06	<input checked="" type="checkbox"/> NGS graphic tablet Draw Master 20x15cm USE	45	342.36	100	01/06/2011	<input checked="" type="checkbox"/>
07	<input checked="" type="checkbox"/> PINNACLE Cubasis VST 5.0	65	237.96	70	01/07/2013	<input checked="" type="checkbox"/>
09	<input checked="" type="checkbox"/> SONY T2XP/S Centr1.2G 512M 60G DVD±RW	15	2632.56	100	01/09/2013	<input checked="" type="checkbox"/>
10	<input type="checkbox"/> WACOM Cintiq graphic tablet 15.1TFT Monit 4DVi	63	59.00	80	01/10/2013	<input checked="" type="checkbox"/>
13		87	1910.28	50	01/01/2013	<input type="checkbox"/>

## Dynamic screens

Screen sections are now dynamic, allowing controls or entire screen to be added or removed at runtime using the DISPLAY UPON SCREEN and DESTROY syntax.

The UPON target can be a screen section, using the following syntax

```
display screen1 upon screen2
```

or can be any of its group levels, allowing creation of child controls, using the syntax

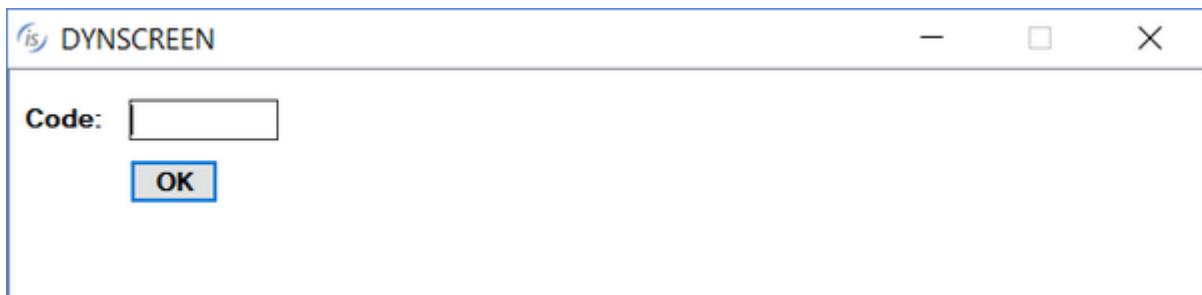
```
display screen1 upon screen2-group-2
```

The following code displays a screen with only a push-button, and populates it at runtime using DISPLAY statements, before accepting user input.

```
program-id. dynscreen.
working-storage section.
77 key-status special-names crt status pic 9(4).
    88 exit-pushed value 27.
77 w-code pic x any length.
screen section.
01 screen1.
    05 push-button ok-button
        line 4 col 9, size 6 cells.
procedure division.
main.
*display a window
    display standard graphical window.
*display the initial screen
    display screen1.
*add a label
    display label upon screen1
        line 2, col 2, size 6 cells
        title "Code:".
*add an entry-field
    display entry-field upon screen1
        line 2, col 9, size 10 cells
        value w-code.
*accept the screen
    perform until exit-pushed
        accept screen1
            on exception
                continue
            end-accept
        end-perform.
    destroy screen1.
```

The result of running the code above is shown on Figure 7, *Dynamic screen*

**Figure 7.** Dynamic screen



## Other enhancements

The message-box control now plays a corresponding notification sound based on the message type, more closely adhering to the Microsoft© Windows© standard.

## Framework improvements

The new release features a new modeless print preview window. Prior to version 2018R2 print preview windows were modal.

The new audit feature allows the auditing of the I/O processing taking place in the application, without requiring code changes.

### Print Preview

isCOBOL Print Preview is now a modeless window allowing multiple simultaneous preview windows. The user can switch between opened preview windows by clicking on them.

### Audit feature

The audit feature builds on top of the I/O Trigger file technology, allowing COBOL developers to add logging on all I/O operations without source code modifications.

The isCOBOL 2018R2 GUI tool provides an easy way to configure the auditing features - specifying users, operations, and files on which auditing is triggered. Auditing can be configured to trigger on all users accessing one or more files, or on specific operations such as delete operations executed from users on a specific file.

The auditing feature is written in COBOL and provided as source code, to be easily customized to any environment and use case.

## Compatibility improvements

Additional ESQL syntax supported in Pro\*COBOL is now supported in isCOBOL, simplifying migration from Pro\*COBOL to JDBC.

New syntax, library routines and configuration properties are supported to enhance compatibility with other COBOL dialects.

### New ESQL syntax:

- Support for ESQL LOCK TABLE statement to acquire a lock on a table or portion of a table.

Code snippet:

```
exec sql
  lock table customers
    in row exclusive mode
    nowait
end-exec.
```

- The FOR clause and the use of array items (OCCURS) without an index is now supported on several ESQL statements. The runtime repeats a statement multiple times when an OCCURS data item is used among host variables. For example, the following code:

```
working-storage section.
77 ins-values pic x(10) occurs 4.
procedure division.
  move "aaa" to ins-values(1).
  move "bbb" to ins-values(2).
  move "ccc" to ins-values(3).
  move "ddd" to ins-values(4).
  exec sql
    for 3
      insert into tbl1 (column1) values (:ins-values)
  end-exec.
```

will cause the runtime to repeat the INSERT statement 3 times, using ins-values(1), ins-values(2) and ins-values(3) respectively. Without the FOR clause, the statement would be executed 4 times, one for each item in ins-values.

- The RETURNING clause is now supported, allowing retrieval of updated values after a statement has been executed, for example:

```
exec sql
  insert into emptbl
    (empno,
     ename,
     deptno
    )
  values
    ('12', 'John Doe', 'dep1')
    returning empno, ename, deptno into
      :new_emp_number, :new_emp_name, :new_dept
  end-exec.
```

- The INTO clause of SELECT and FETCH ESQL statements can now be bound to an OCCURS data item, for example:

```
working-storage section.
77 col-val pic x(10) occurs 3.
procedure division.
  exec sql
    select column1 into :col-val from tbl1
  end-exec.
```

The above statement will read 5 records from table tbl1 and store the values of field column1 in col-val(1), col-val(2), col-val(3).

## Improved COBOL compatibility

isCOBOL now provides even better compatibility with other COBOL dialects.

- Properties in CLASS-ID programs are now supported. The CLASS-ID program sets the property as it would do with a standard variable, for example:

```
identification division.  
Class-id. myClass.  
identification division.  
factory.  
working-storage section.  
01 myProp pic 9(10) comp property.  
procedure division.  
identification division.  
method-id. myMethod  
procedure division.  
main.  
    move 2 to myProp.
```

- Legacy programs and CLASS-ID programs that reference the class above can also reference the property in the REPOSITORY paragraph, and use it as a standard COBOL variable, for example:

```
configuration section.  
repository.  
    class myClass as "myClass"  
        property myProp.  
    procedure division.  
    main.  
        display myProp of myClass.
```

- ACTIVE-CLASS syntax is now supported, allowing you to identify a specific instance of the current class or one of its subclasses. Code snippet:

```
method-id. getInstance as "getInstance".  
working-storage section.  
77 cls-instance object reference active-class.  
procedure division returning cls-instance.  
main.  
    invoke self "new" giving cls-instance.
```

- New Micro Focus® directives are now supported:

```
$SET INDD"<filename>"  
$SET OUTDD"<filename [recsize] [filetype]>"
```

To enable you to map the system input (SYSIN) and system output (SYSOUT) to disk files. Each program can use different disk files. This is explained in detail later in this document.

## New library routine

A new library routine has been implemented to provide better compatibility with RM/COBOL®:

- C\$WRU returns the name of the calling program.

The following code snippet shows the usage of the C\$WRU library routine

```
01 WHO-CALLED-ME .
  05 THE-CALLING-PROGRAM PIC X(30)  VALUE SPACES .
  05 THE-CALLING-LINE      PIC S9(6)  BINARY .
  05 THE-LINE-NUM         PIC S9(02) BINARY .
PROCEDURE DIVISION .
  CALL 'C$WRU' USING THE-CALLING-PROGRAM
                  THE-CALLING-LINE
                  THE-LINE-NUM .
```

## New configuration property

The new *iscobol.memory.alpha\_edited=true* configuration property has been implemented to manage the VALUE clause of alphanumeric edited items in compatibility with Micro Focus®, AcuCOBOL-GT® and RM/COBOL®.

## isCOBOL Compiler Enhancements

isCOBOL Evolve 2018 R2 implements new compiler options and new syntax to better support COBOL applications that use DBCS (Double Byte Character Strings) such as Japanese, Chinese and Korean without Unicode encoding.

The Compiler can now automatically compile classes used by the program being compiled.

### New compiler options

- -ccbas to count bytes instead of characters for fixed (aka ANSI) source code

This option helps when compiling source files that contain DBCS text. By default, the isCOBOL compiler counts characters in order to determine the text positions of the various areas of the Fixed (ANSI) COBOL source format, while Asian compilers count the bytes instead. Source files written for Asian COBOLs may not cleanly compile without the -ccbas option.

- -cndbc to use the DBCS instead of Unicode in PIC N without the USAGE NATIONAL clause.

A data item such as:

```
77 n-item pic n(10) .
```

stores data in UTF-16 Big Endian by default. When compiling the program with -cndbc data will be stored using the current encoding. If both Unicode and DBCS data items are needed in the same application, they can be declared as follows:

```
77 dbcs-item   pic n(10) .
77 unicode-item pic n(10) USAGE NATIONAL .
```

### Automatic class compilation

When you compile COBOL programs (PROGRAM-ID) or COBOL classes (CLASS-ID) that reference additional COBOL classes (CLASS-ID), the Compiler now automatically compiles the referenced source code.

For example, considering the following three source code files:

Prog.cbl:

```
identification division.  
program-id. Prog.  
environment division.  
configuration section.  
repository.  
    class Class2 as "Class2".  
procedure division.  
main.  
    Class2:>method1().  
    goback.
```

Class1.cbl:

```
identification division.  
class-id. Class1 as "Class1".  
identification division.  
factory.  
procedure division.  
identification division.  
method-id. method1 as "method1".  
procedure division.  
main.  
* do something here  
end method.  
end factory.
```

Class2.cbl:

```
identification division.  
class-id. Class2 as "Class2" inherits Class1.  
environment division.  
configuration section.  
repository.  
    class Class1 as "Class1".  
identification division.  
factory.  
procedure division.  
*add some methods here  
end factory.
```

When compiling Prog.cbl, the Class1.cbl and Class2.cbl source files are automatically compiled, both when compiling from the IDE or from the command line.

### SYSIN and SYSOUT mapping

Two new compiler properties have been added to allow you to map the system's standard input (SYSIN) and standard output (SYSOUT) to disk files.

```
iscobol.compiler.indd=<filename>  
iscobol.compiler.outdd=<filename [recsize] [filetype]>
```

For example, let's consider using the following compiler configuration properties:

```
iscobol.compiler.indd=input.txt  
iscobol.compiler.outdd=output.txt
```

If a program is compiled with those properties, it will read a line from the file input.txt instead of accepting the user input when it performs an ACCEPT dest-item FROM SYSIN. When the program performs a DISPLAY ... UPON SYSOUT, it will write a line to the file output.txt instead of printing on the console.

To allow programs to use program-specific input and output files, these' two properties can also be used as compiler directives inside the source code. For example:

```
$SET INDD "input-prog1.txt"  
$SET OUTDD "output-prog1.txt"  
program-id. prog1.
```

## isCOBOL Debugger

The isCOBOL Debugger has greatly enhanced, with new features aimed at increasing developer productivity, such as a new debugger command, easier code navigation and more.

### New debugger command

The isCOBOL Debugger has been enhanced with the new feature “Run to next program”, using the command PROG. When the PROG command is used, the Debugger continues execution of the current program, stopping at the beginning of the next COBOL program. This is especially useful when the caller program is not a COBOL program, such as a Java or C program, allowing the debugger to stop at the first COBOL program called.

For example, by typing PROG in the command window, pressing the button on the toolbar shown in Figure 8, *Debugger command PROG*, or choosing the option from the menu, the COBPROG2 will be executed and the Debugger will stop when running COBPROG3, which is the next program called by the main Java source.

**Figure 8.** Debugger command PROG

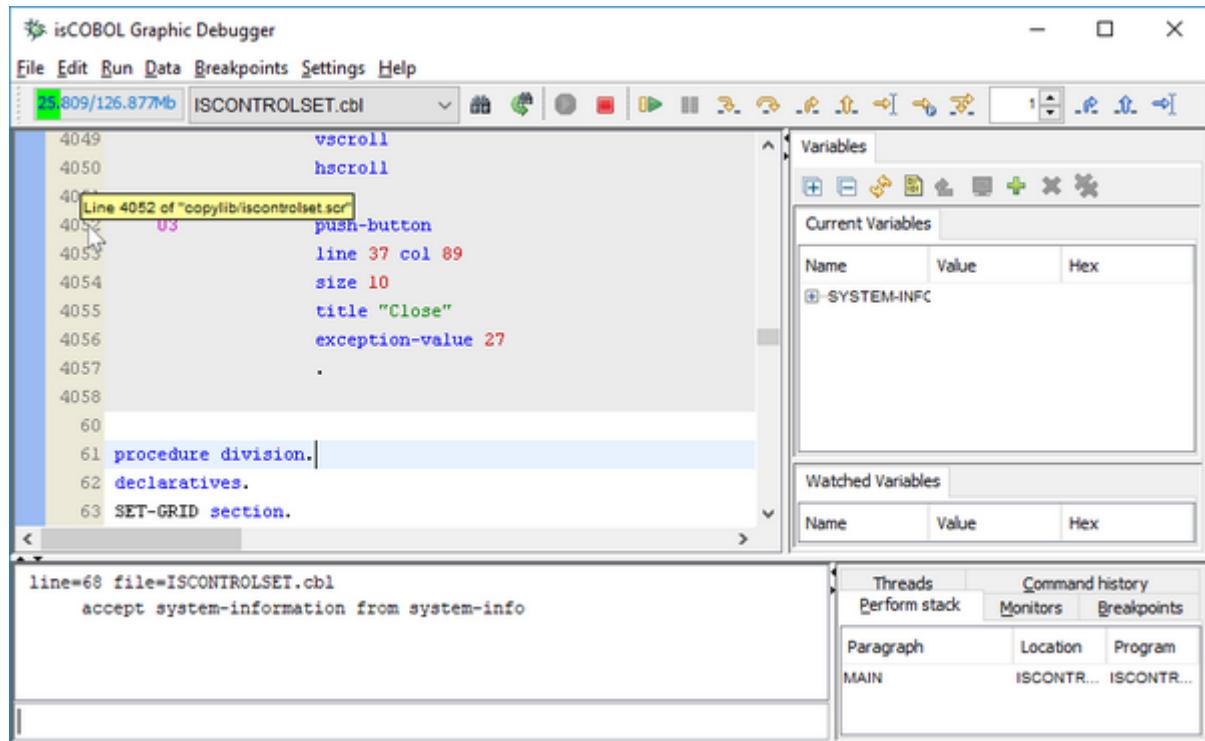
The screenshot shows the isCOBOL Graphic Debugger interface. The main window displays the COBPROG2.cbl source code. Line 17 is highlighted with a yellow background and contains the command `DISPLAY "Inside COBOL program 'COBPROG2'"`. The right side of the interface features several panes: Variables, Current Variables, Watched Variables, Threads, Command history, Paragraph, Location, and Program. The Command history pane at the bottom left shows the command `test CALL with 3 parameters` followed by two entries for line 16 and line 17, both from COBPROG2.cbl, with the same displayed message.

```
4 IDENTIFICATION DIVISION.  
5 PROGRAM-ID. COBPROG2.  
6  
7 WORKING-STORAGE SECTION.  
8 77 ENV-VALUE PIC X(50).  
9  
10 LINKAGE SECTION.  
11 01 WL-PAR-01    pic x(20).  
12 01 WL-PAR-02    pic x(20).  
13 01 WL-PAR-03    pic 9(4).  
14  
15 PROCEDURE DIVISION USING WL-PAR-01, WL-PAR-02, WL-PAR-03.  
16 MAIN.  
17 DISPLAY "Inside COBOL program 'COBPROG2'"  
18  
19     DISPLAY "Parameter 1"  
20     DISPLAY WL-PAR-01  
21     DISPLAY "Parameter 2"  
22     DISPLAY WL-PAR-02  
23     DISPLAY "Parameter 3"  
24     DISPLAY WL-PAR-03  
25  
test CALL with 3 parameters  
line=16 file=COBPROG2.cbl  
line=17 file=COBPROG2.cbl  
    DISPLAY "Inside COBOL program 'COBPROG2'"
```

### Hint on line column

In large programs with many copybook files, sometimes it's difficult to determine which file contains the source code being debugged. With the new debugger, a hint will appear when hovering the mouse pointer over the line number column, located on the left pane of the Debugger window, displaying the file path and line number of the highlighted line. Figure 9, *Debugger hint on line column*, shows the feature in action.

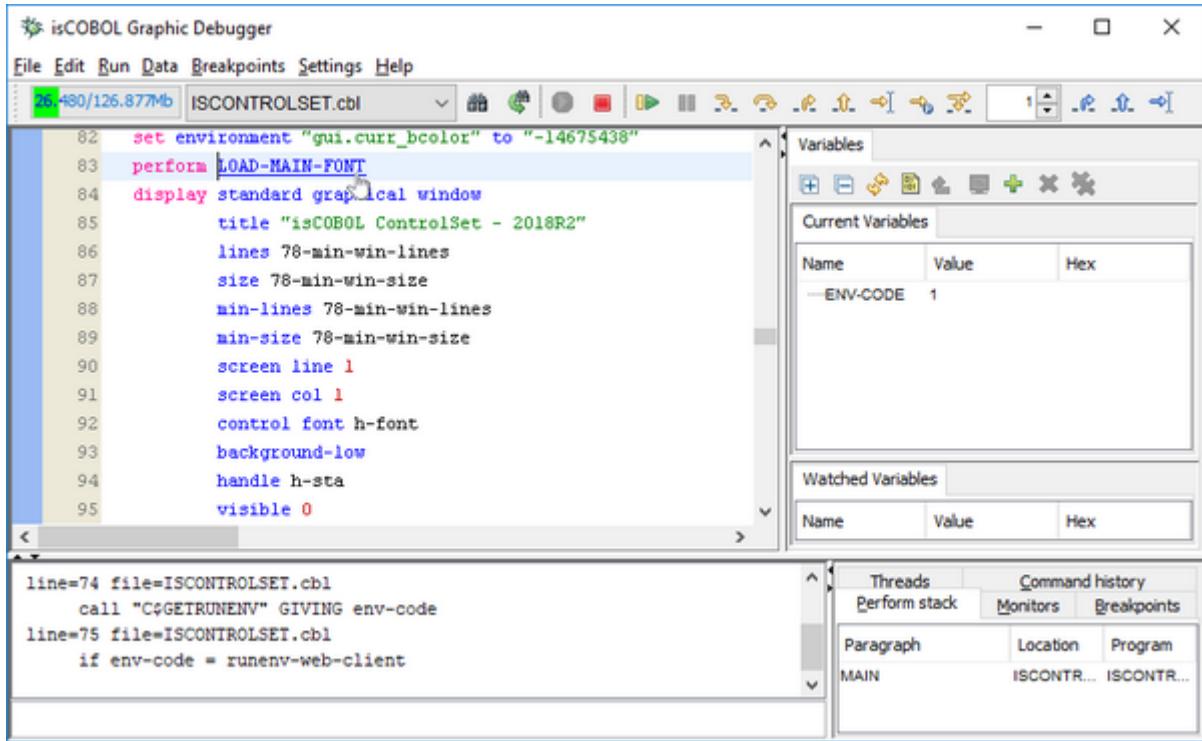
**Figure 9.** Debugger hint on line column



## Hyperlink

To simplify jumping to a given paragraph or variable definition, the hyperlink feature has been introduced. When hovering the mouse pointer on a paragraph name or a variable name, the name will be underlined and the mouse cursor will change to hand shape point, as shown in Figure 10, *Debugger hyperlink*. Left clicking the name will cause the debugger to display the definition.

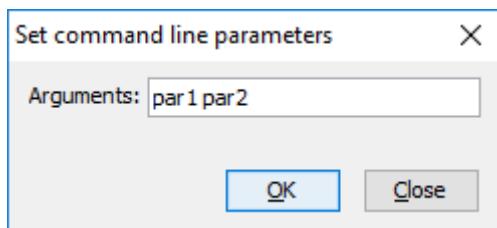
**Figure 10.** Debugger hyperlink



## Command line parameters

The 'Set command line parameters' values are now saved in the Debugger session file (.isd), making it easier to debug a program with CHAINING parameters in different debugging sessions, as shown in Figure 11, *Debugger Set command line parameters*.

**Figure 11.** Debugger Set command line parameters



## isCOBOL EIS

The isCOBOL EIS features have been improved in several areas, such as the HTTPHandler class, webDirect routines, and Stream2Wrk utility.

### HTTPHandler improvements

isCOBOL EIS now allows access to key Servlet objects in both COBOL Servlets and webDirect applications.

The HTTPHandler class provides the following new methods:

- `HTTPHandler:>getRequest()`

- `HTTPHandler:>getResponse()`
- `HTTPHandler:>getSession()`

These methods return the instance of the HTTP Request, Response and Session respectively. For example, these objects can be used to retrieve the IP address of the end user in your Servlet, using the following code:

```

repository.
  class HTTPHandler as "com.iscobol.rts.HTTPHandler"
  class HTTPRequest as "javax.servlet.ServletRequest"

  .
  working-storage section.
  77 servlet-request object reference HTTPRequest.
  77 client-ip pic x any length.
  linkage section.
  77 http-handler object reference HTTPHandler.
  procedure division using http-handler.
    set servlet-request to http-handler:>getRequest()
      as HTTPRequest.
    set client-ip to servlet-request:>getRemoteAddr() .

```

## webDirect improvements

The webDirect routine `WD2$SESSION` has been enhanced with new properties that return information about the servlet context and the HTTP session of the servlet managing the COBOL application. They are:

- `iscobol.wd2.servletcontext.name`
- `iscobol.wd2.servletcontext.realpath`
- `iscobol.wd2.servletcontext.path`
- `iscobol.wd2.servletcontext.serverinfo`
- `iscobol.wd2.servletcontext.majorversion`
- `iscobol.wd2.servletcontext.minorversion`
- `iscobol.wd2.httpsession.id`
- `iscobol.wd2.httpsession.creationtime`

For example, to retrieve the real path of an application deployed in a servlet container, the following code can be used:

```

working-storage section.
copy "iscobol.def".
 77 real-path pic x any length.
procedure division.
  call "wd2$session" using wd2-get-session-value
    "iscobol.wd2.servletcontext.realpath"
    real-path.

```

Also, the rendering of controls in webDirect has been upgraded to improve screen section display.

## Stream2Wrk improvements

The Stream2Wrk utility provides a new option to specify the name of the root element when processing JSON streams that have no root element.

For example, the following JSON:

```
{  
  "name": "John",  
  "age": 30,  
  "cars": [ "Ford", "BMW", "Fiat" ]  
}
```

would generate the following working-storage structure:

```
01 json2wrk identified by ''.  
 03 name identified by 'name'.  
    05 name-data pic x any length.  
 03 age identified by 'age'.  
    05 age-data pic x any length.  
 03 cars identified by 'cars' occurs dynamic capacity cars-count.  
    05 cars-data pic x any length.
```

To change, for example, the automatically generated “json2wrk” name created by Stream2Wrk to “allcars”, the following command line can be used:

```
$ stream2wrk json yourfile.json -r allcars
```

The result will be:

```
01 allcars identified by ''.  
 03 name identified by 'name'.  
    05 name-data pic x any length.  
 03 age identified by 'age'.  
    05 age-data pic x any length.  
 03 cars identified by 'cars' occurs dynamic capacity cars-count.  
    05 cars-data pic x any length.
```

## isCOBOL Evolve

---

# isCOBOL 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 webClient. 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 webDirect can both create web applications, but they differ in how this is accomplished. webDirect allows the use of external CSS stylesheets and javascript files to spice 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*.

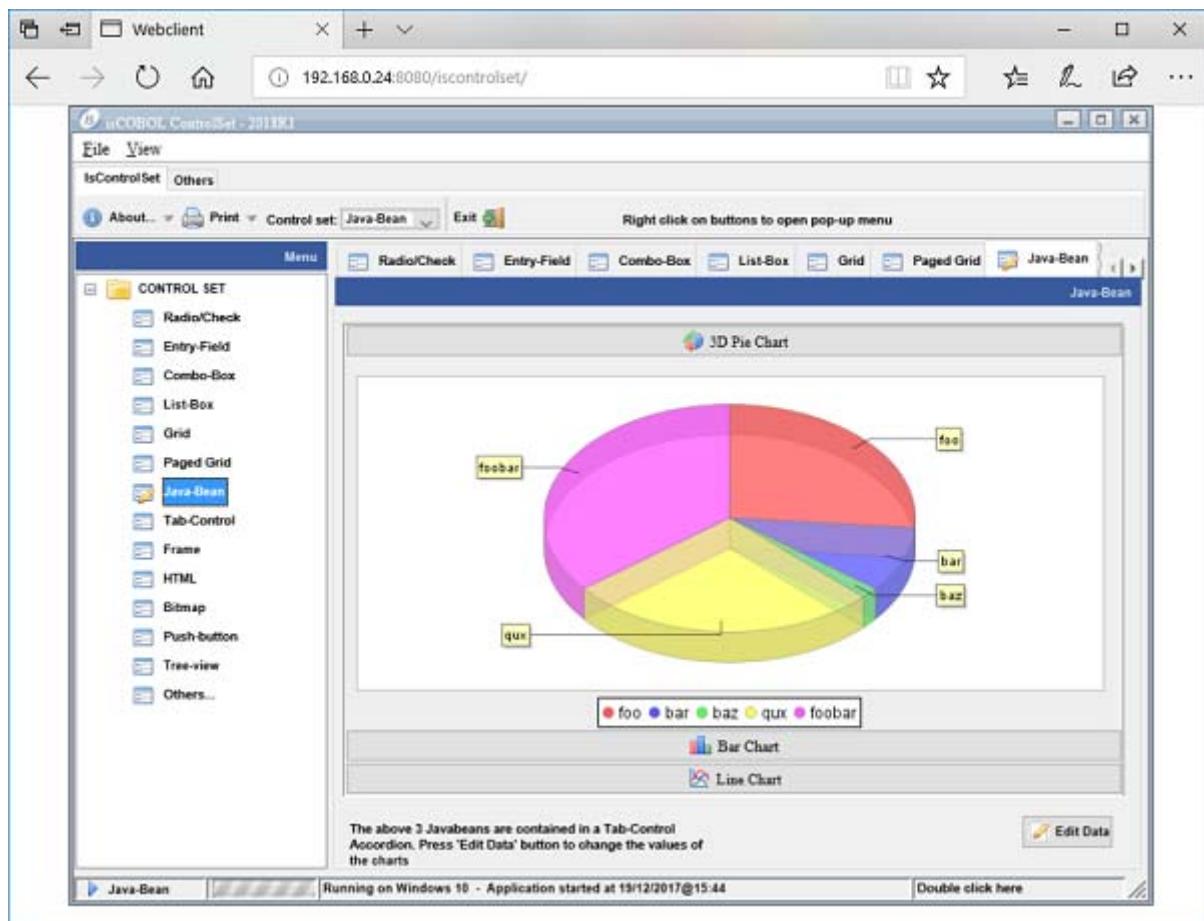
**Figure 1.** Configuration page

The screenshot shows the 'WebClient console' window with the URL '192.168.0.24:8080/admin/#/config/swing/iscontrolset'. The navigation bar includes 'WebClient', 'Dashboard', 'Configuration', 'Logs', 'Exit', and 'Logout'. Below the navigation bar, there's a 'Sessions' section with a 'Back' button and a search field. A card for 'iscontrolset' shows it is 'Running'. A 'Disable' button is also present. The main area is titled 'Configuration' with 'Reset' and 'Apply' buttons. The configuration fields are:

Enabled	<input checked="" type="checkbox"/> ON
Home Folder	<input type="button" value="..."/> \${user.dir}
Icon	<input type="button" value="..."/> \${webclient.rootDir}/iscobol.png
Security Module Name	<input type="button" value="..."/> NONE
Security Module Class Path	<input type="button" value="+"/>
Name	<input type="button" value="..."/> iscontrolset
isCOBOL Server address	<input type="button" value="..."/> 127.0.0.1
isCOBOL Server port	<input type="button" value="..."/> 10999
Program name and arguments	<input type="button" value="..."/> ISCONTROLSET

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

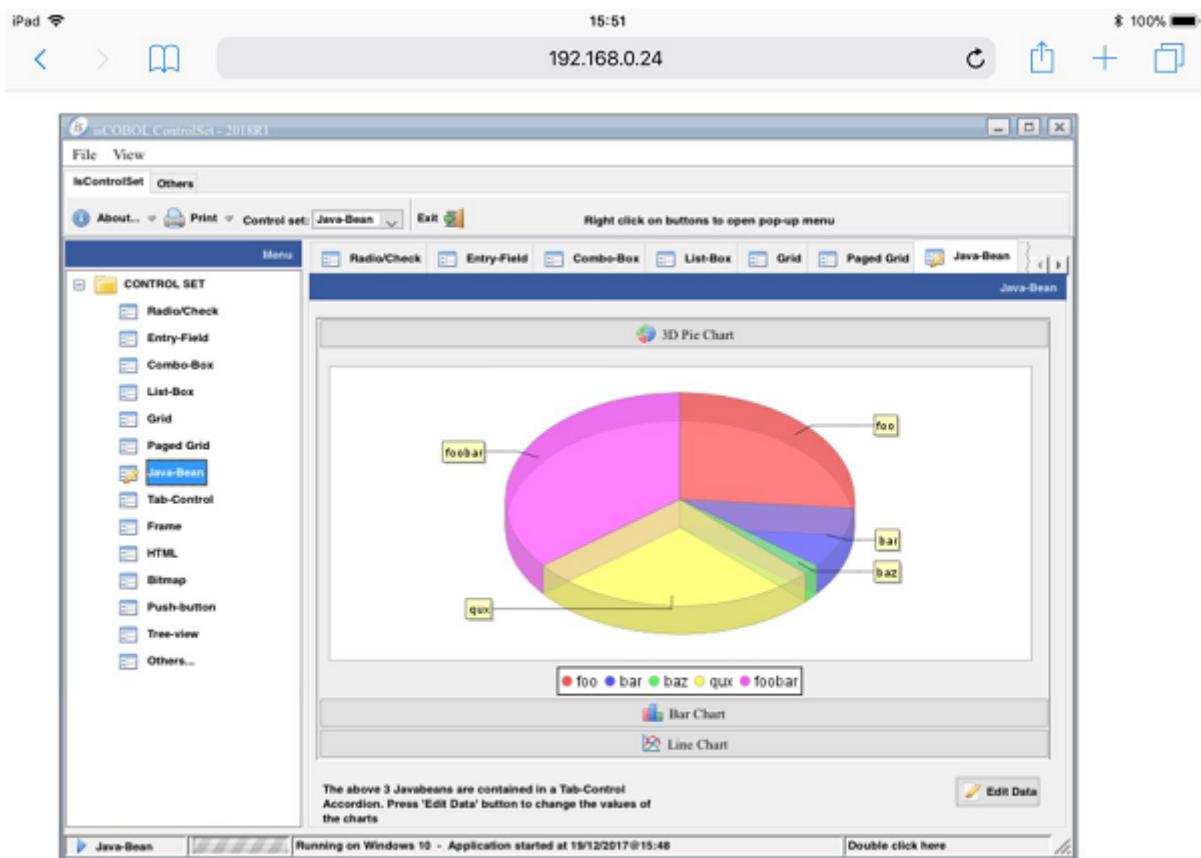


Figure 4, *Administrator view of running applications*, shows how administrators can monitor running processes on the webClient server, and view or act on the user's screen, as shown in Figure 5, *Administrator view of user session*.

**Figure 4.** Administrator view of running applications

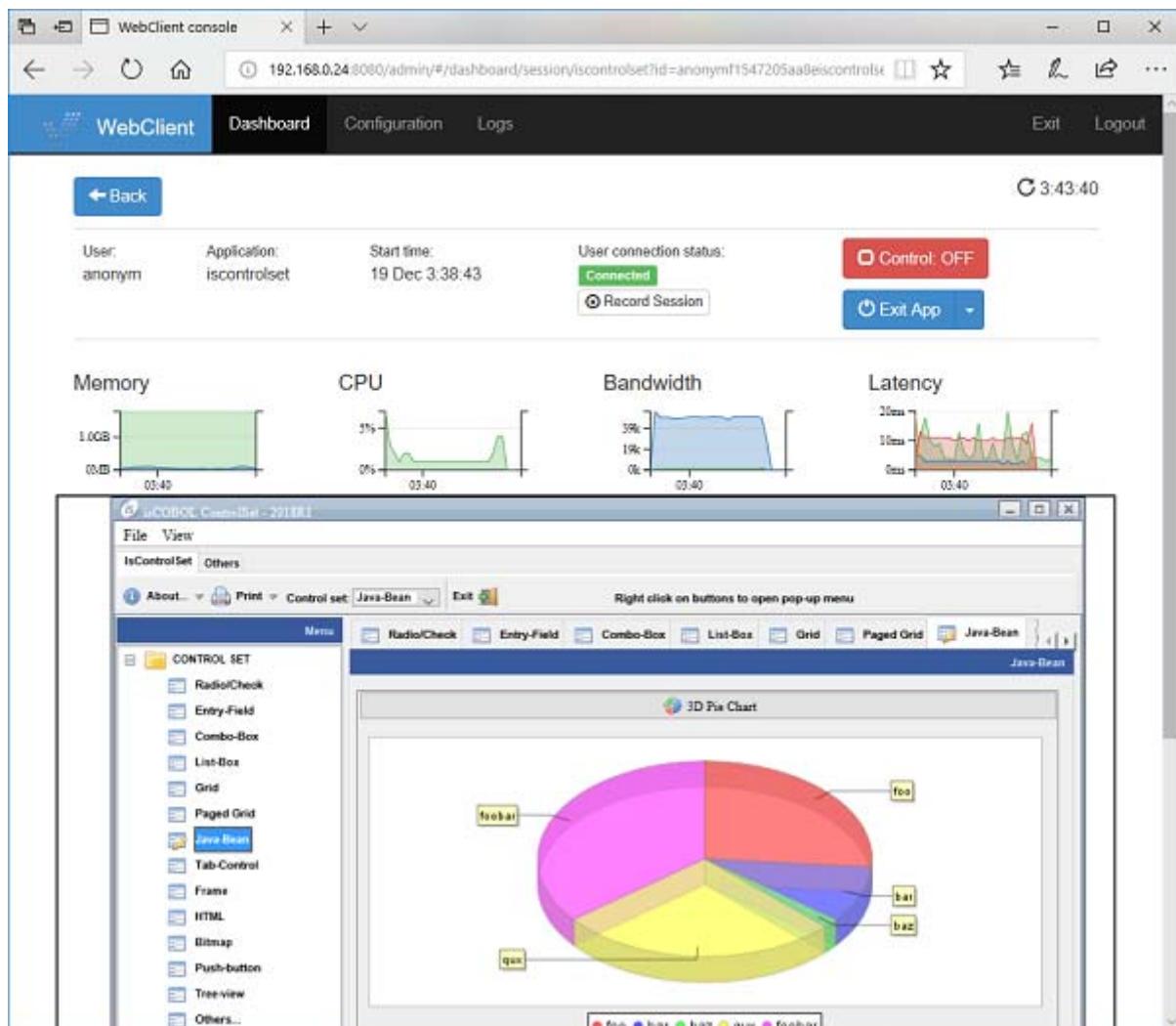
The screenshot shows the WebClient console interface for managing application sessions. At the top, there's a navigation bar with tabs for 'WebClient', 'Dashboard', 'Configuration', and 'Logs', along with 'Exit' and 'Logout' buttons. Below the navigation bar, the URL is shown as 192.168.0.24:8080/admin/#/dashboard/overview/iscontrolset. The main content area is titled 'iscontrolset /iscontrolset Running' with a 'Disable' button. The 'Running sessions' section lists two sessions:

No. (Id)	User	IP	Start time	Client status	Metrics (min   avg   max)	Bandwidth (min   avg   max)	Latency (min   avg   max)	Actions
1	anonim	192.168.0.229	19 Dec 3:38:43	Connected Record Session	MEM: 48MB (0 62 105) CPU: 2% (0 2 6)	IN: 1k/s (0 1 1) OUT: 50k/s (0 47 55)	E 15ms (0 17 29) PING 4ms (0 8 19)	<a href="#">View</a> <a href="#">Shutdown</a>
2	admin	192.168.0.24	19 Dec 3:36:20	Connected Record Session	MEM: 49MB (0 42 74) CPU: 2% (0 2 6)	IN: 1k/s (0 1 1) OUT: 42k/s (0 41 42)	E 14ms (0 16 22) PING 4ms (0 7 16)	<a href="#">View</a> <a href="#">Shutdown</a>

The 'Finished sessions' section lists two completed sessions:

No. (Id)	User	IP	Start time	End time	Status	Actions
1	admin	192.168.0.24	19 Dec 15:41:20	19 Dec 15:41:43 (0 min 23 sec)	Finished	
2	anonim	192.168.0.229	19 Dec 15:36:46	19 Dec 15:38:40 (1 min 54 sec)	Finished	

**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.

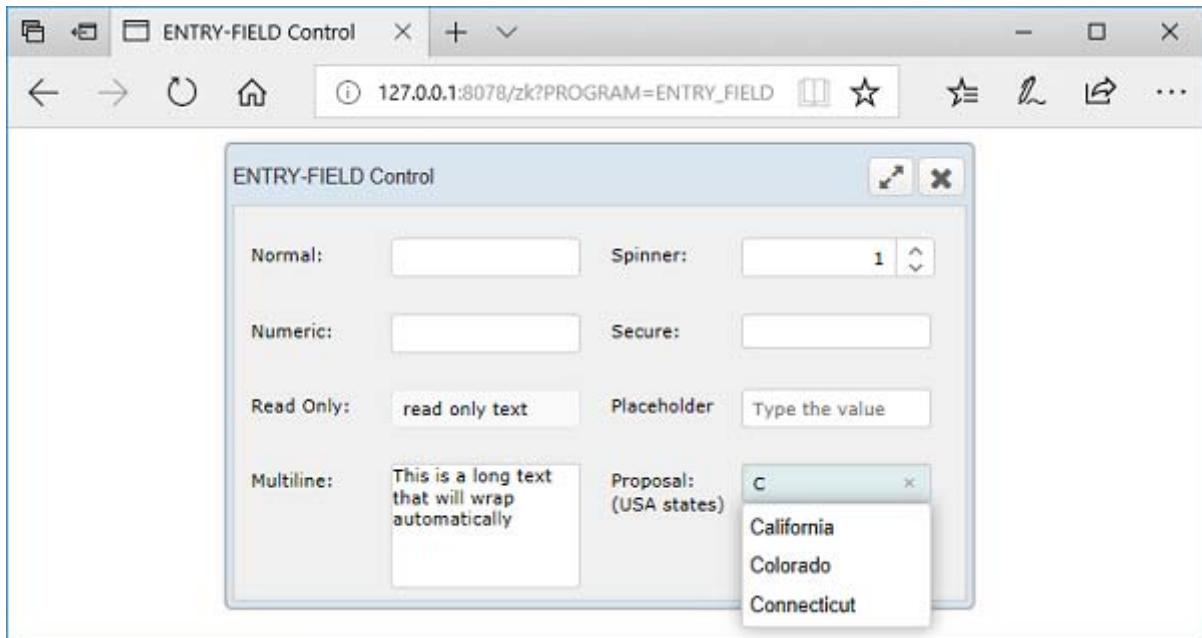
IsCOBOL webDirect now supports the PROPOSAL property on entry-field controls

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 webDirect*.

**Figure 6.** Proposal in webDirect



## 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:

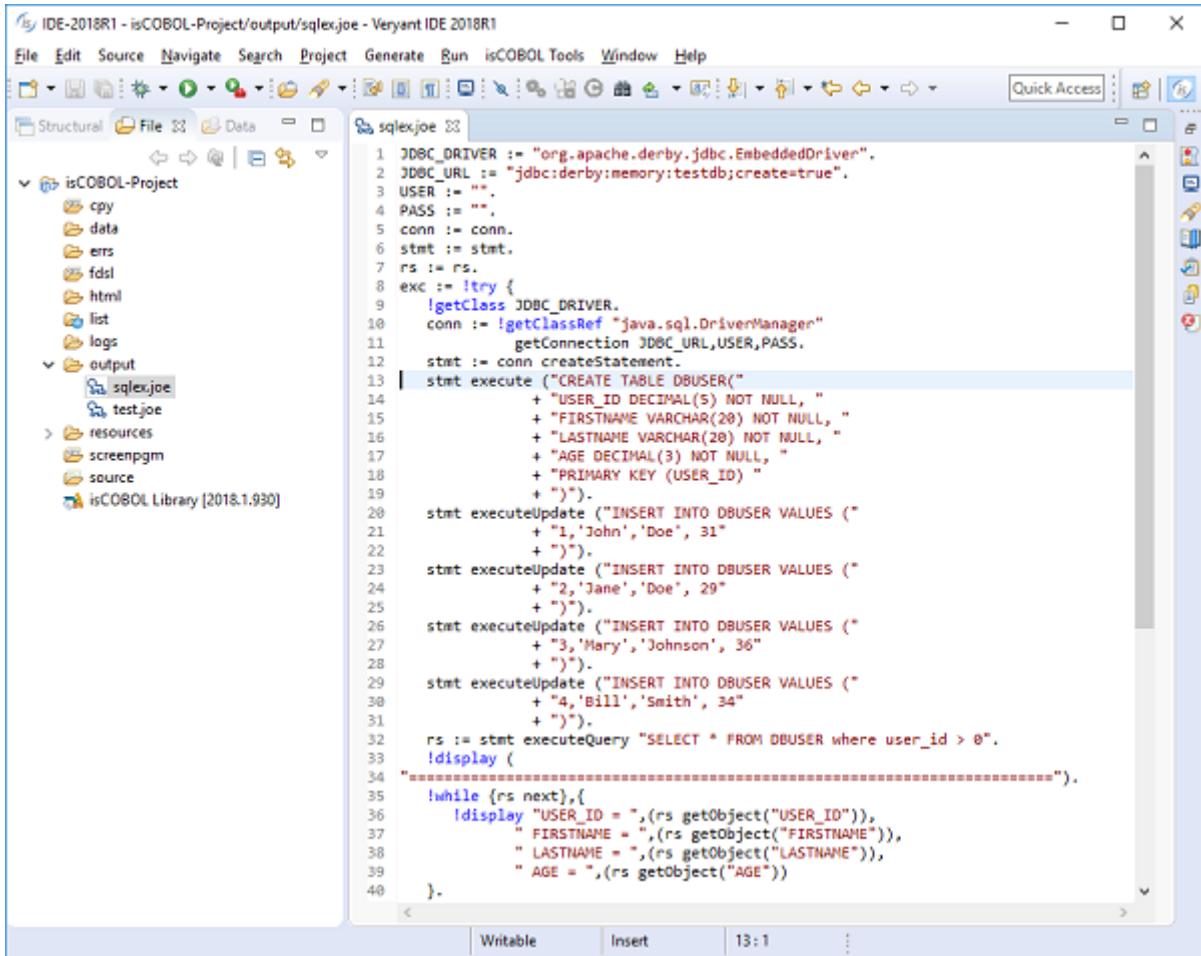
```
/** JOE says Hello */
!display "What's your name?".
VAR := !accept.
!display "Hello ",VAR.
!exit.
```

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



The screenshot shows the isCOBOL IDE 2018R1 interface. The title bar reads "IDE-2018R1 - isCOBOL-Project/output/sqlxjoe - Veryant IDE 2018R1". The menu bar includes File, Edit, Source, Navigate, Search, Project, Generate, Run, isCOBOL Tools, Window, Help. The toolbar has various icons for file operations. The left sidebar shows a project structure with "isCOBOL-Project" expanded, containing "cpy", "data", "errs", "fdsl", "html", "list", "logs", "output" (which contains "sqlxjoe" and "testjoe"), "resources", "screenpgm", and "source". The "sqlxjoe" file is open in the main editor area, displaying Java code for JDBC operations. The code creates a database connection, executes SQL statements to insert data into a table named DBUSER, and then retrieves and displays the results. The code uses JDBC 4.0 features like try-with-resources and PreparedStatement. The status bar at the bottom shows "Writable", "Insert", and the time "13:1".

```

1 JDBC_DRIVER := "org.apache.derby.jdbc.EmbeddedDriver".
2 JDBC_URL := "jdbc:derby:memory:testdb;create=true".
3 USER := "".
4 PASS := "".
5 conn := conn.
6 stmt := stmt.
7 rs := rs.
8 exc := !try {
9     !getClass JDBC_DRIVER.
10    conn := !getClassRef "java.sql.DriverManager"
11        getConnection JDBC_URL,USER,PASS,
12    stmt := conn.createStatement.
13    stmt execute ("CREATE TABLE DBUSER(
14        + "USER_ID DECIMAL(5) NOT NULL, "
15        + "FIRSTNAME VARCHAR(20) NOT NULL, "
16        + "LASTNAME VARCHAR(20) NOT NULL, "
17        + "AGE DECIMAL(3) NOT NULL, "
18        + "PRIMARY KEY (USER_ID) "
19        + ")");
20    stmt executeUpdate ("INSERT INTO DBUSER VALUES (
21        + "1,'John','Doe', 31"
22        + ")");
23    stmt executeUpdate ("INSERT INTO DBUSER VALUES (
24        + "2,'Jane','Doe', 29"
25        + ")");
26    stmt executeUpdate ("INSERT INTO DBUSER VALUES (
27        + "3,'Mary','Johnson', 36"
28        + ")");
29    stmt executeUpdate ("INSERT INTO DBUSER VALUES (
30        + "4,'Bill','Smith', 34"
31        + ");
32    rs := stmt executeQuery "SELECT * FROM DBUSER where user_id > 0".
33    !display (
34        =====
35        !while {rs next},
36            !display "USER_ID = ",(rs getObject("USER_ID")),
37            " FIRSTNAME = ",(rs getObject("FIRSTNAME")),
38            " LASTNAME = ",(rs getObject("LASTNAME")),
39            " AGE = ",(rs getObject("AGE"))
40        }.

```

## 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.

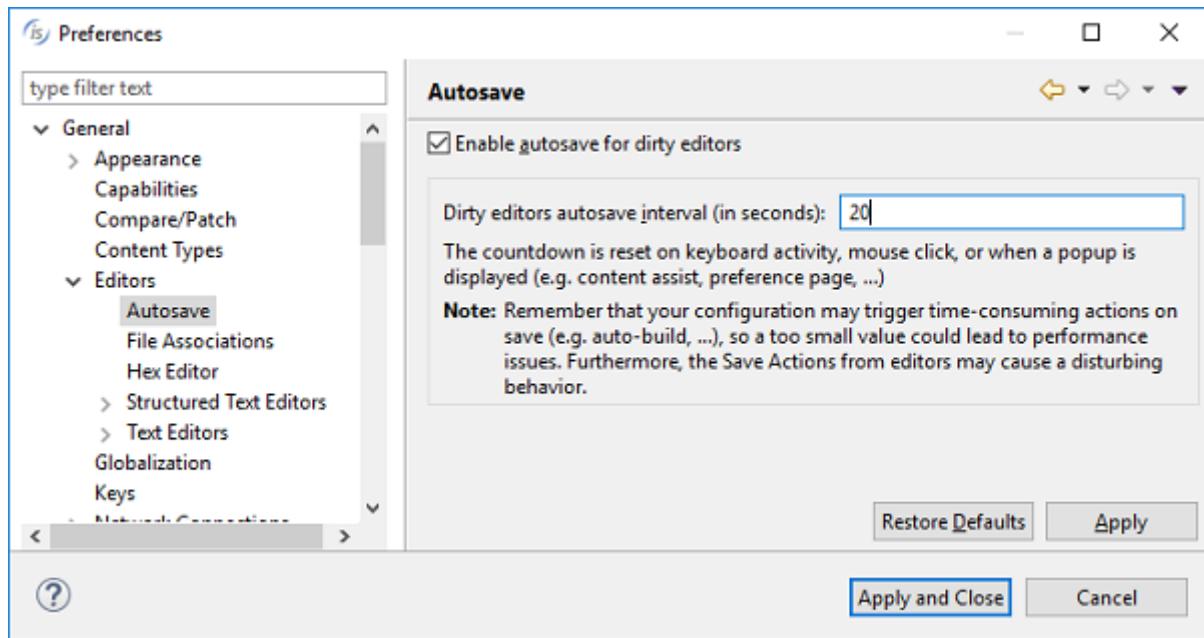
### Eclipse Oxygen

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.

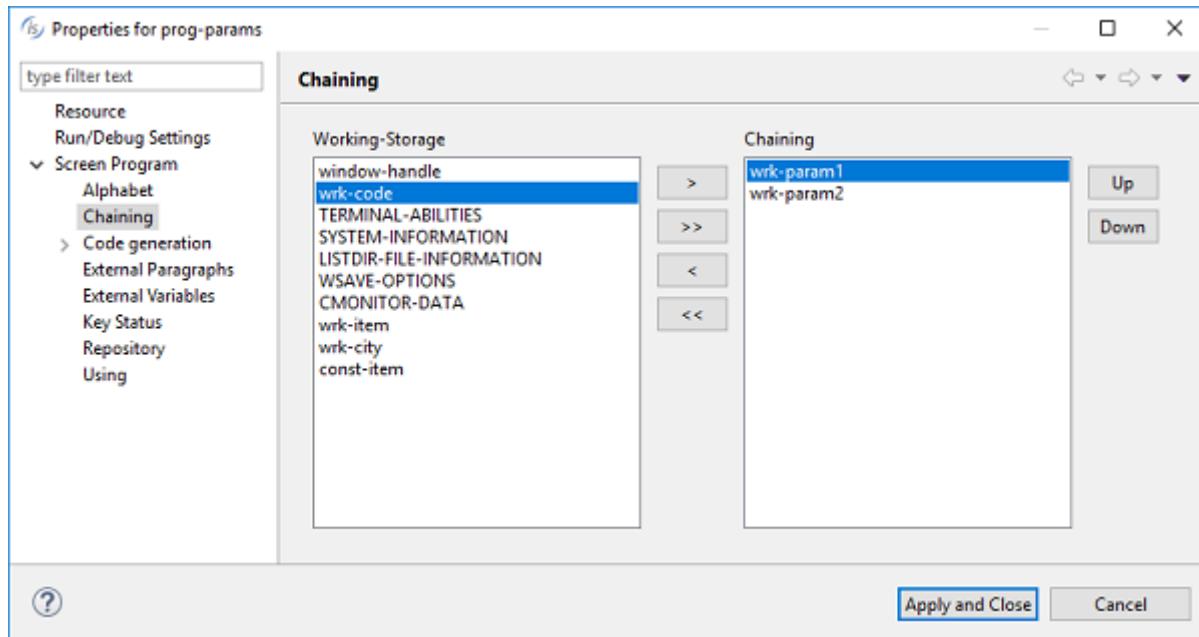
**Figure 8.** Autosave feature



### 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.

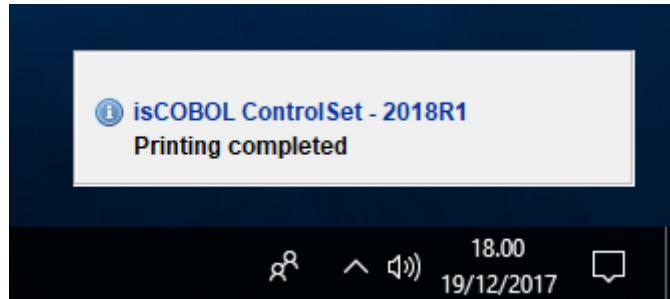
A notification window uses a customizable corner of the desktop to display notifications or status information.

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

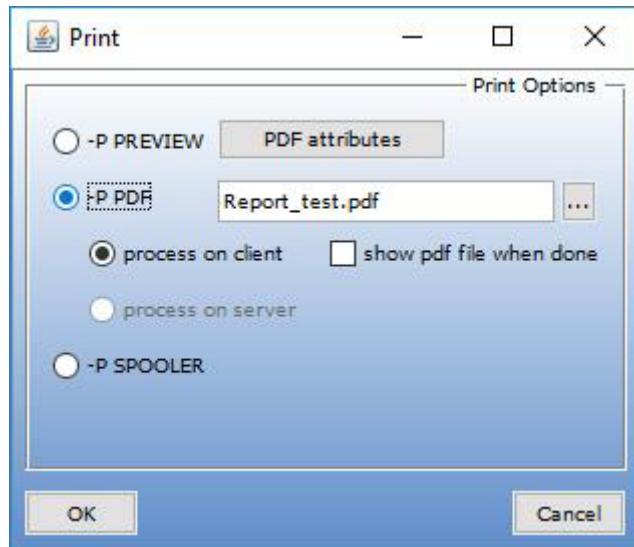
GRADIENT-ORIENTATION specifies the orientation of the gradient effect. Allowed values are: gradient-north-to-south, gradient-northeast-to-southwest, gradient-east-to-west, gradient-southeast-to-northwest, gradient-south-to-north, gradient-southwest-to-northeast, gradient-west-to-east, gradient-northwest-to-southeast.

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 iscobel.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 iscobel-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"
```

# isCOBOL 2017 Release 2 Overview

## Introduction

Veryant is pleased to announce to selected users 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.

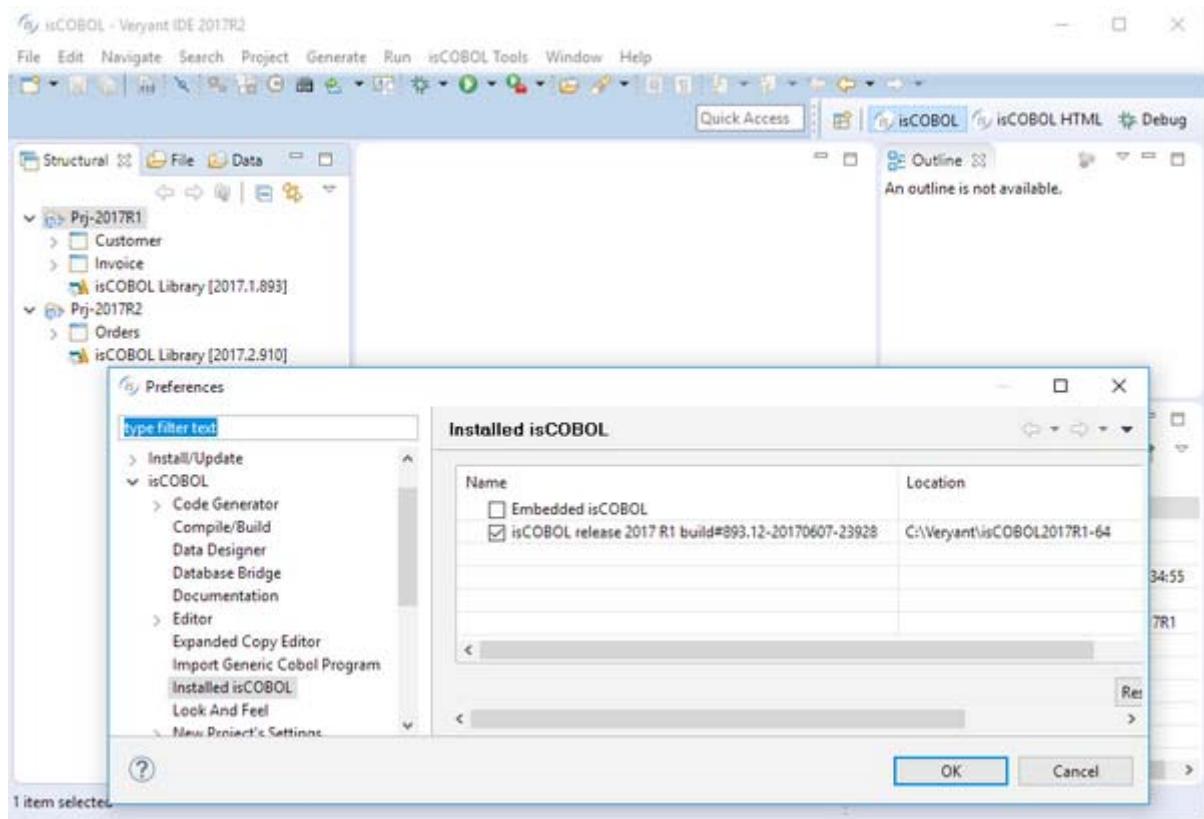
Details on these enhancements and updates are included below.

## isCOBOL IDE Enhancements

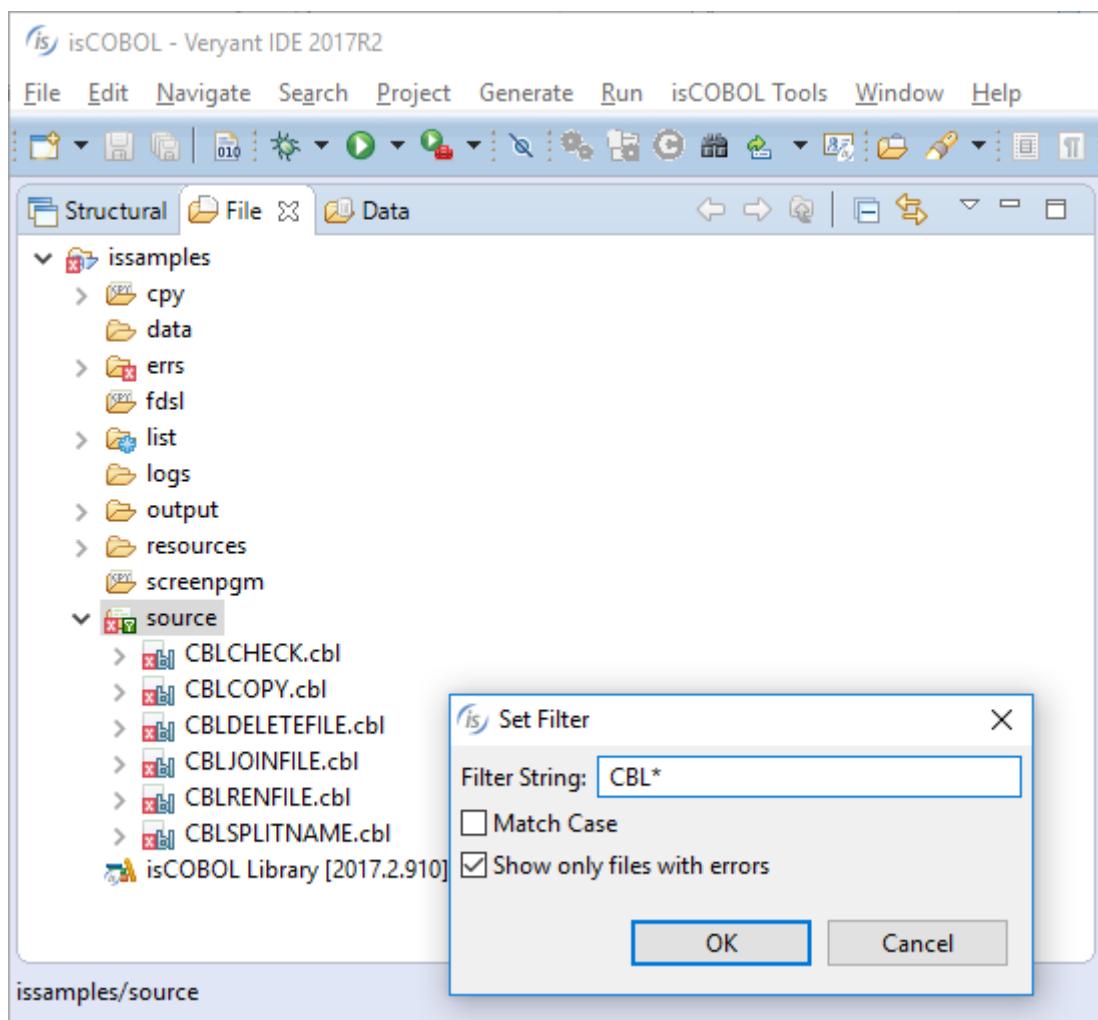
The isCOBOL 2017 R2 IDE now allows developers to set a different isCOBOL compiler version to use while compiling and testing for each project in the workspace, instead of the one contained within the IDE.

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.

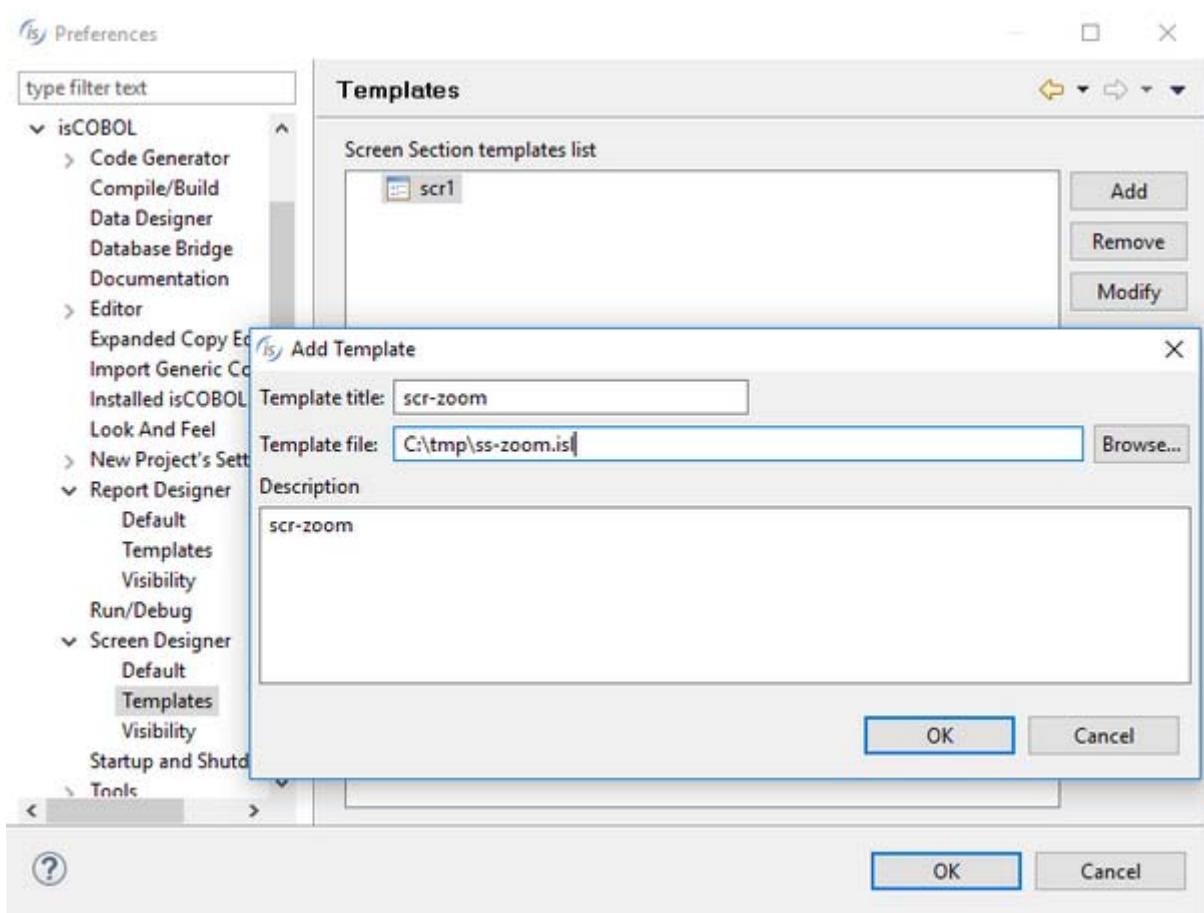
The minimum compatible compiler version is 2017 R1.



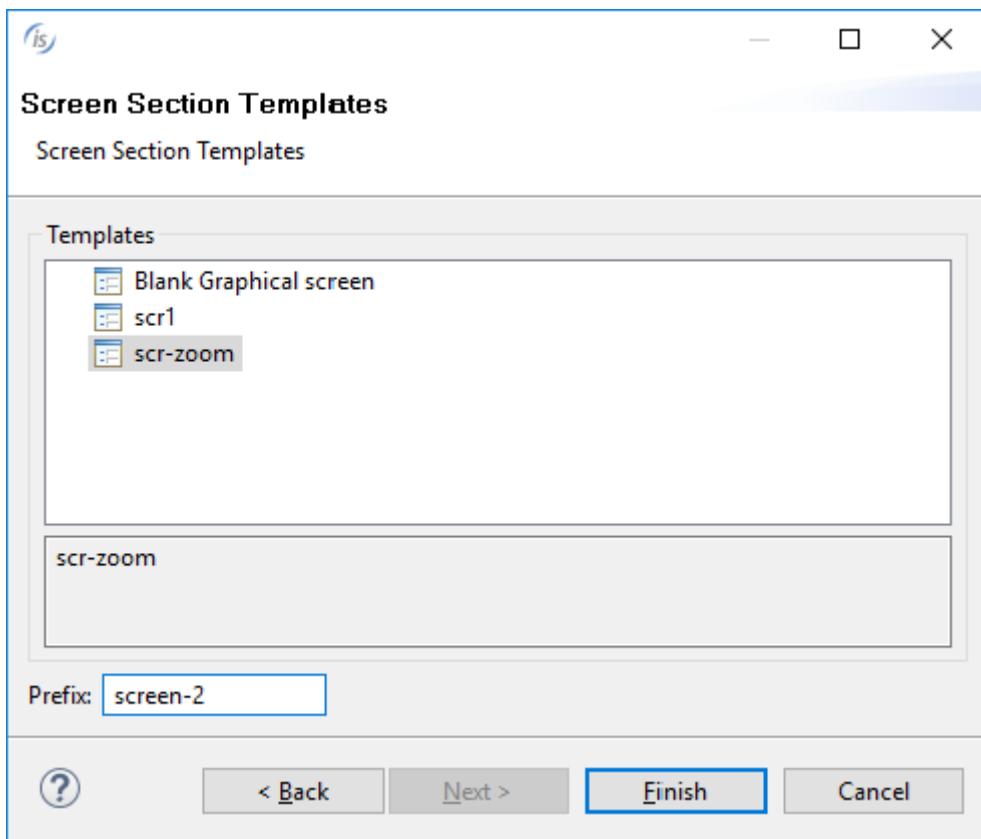
The "Set Filter" feature has been improved by adding the option to show only programs that have compilation errors, making it much easier to find source files that need fixing in very large projects.



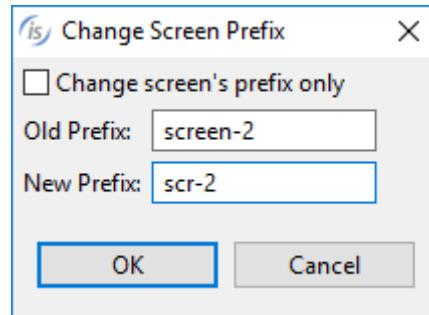
The import Report and Screen features have been improved by allowing the use of custom templates, and by allowing the Report and Screen prefix to be modified when importing the template, enabling teams to set standards when creating new screens and reports in an application. Templates can be set in the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



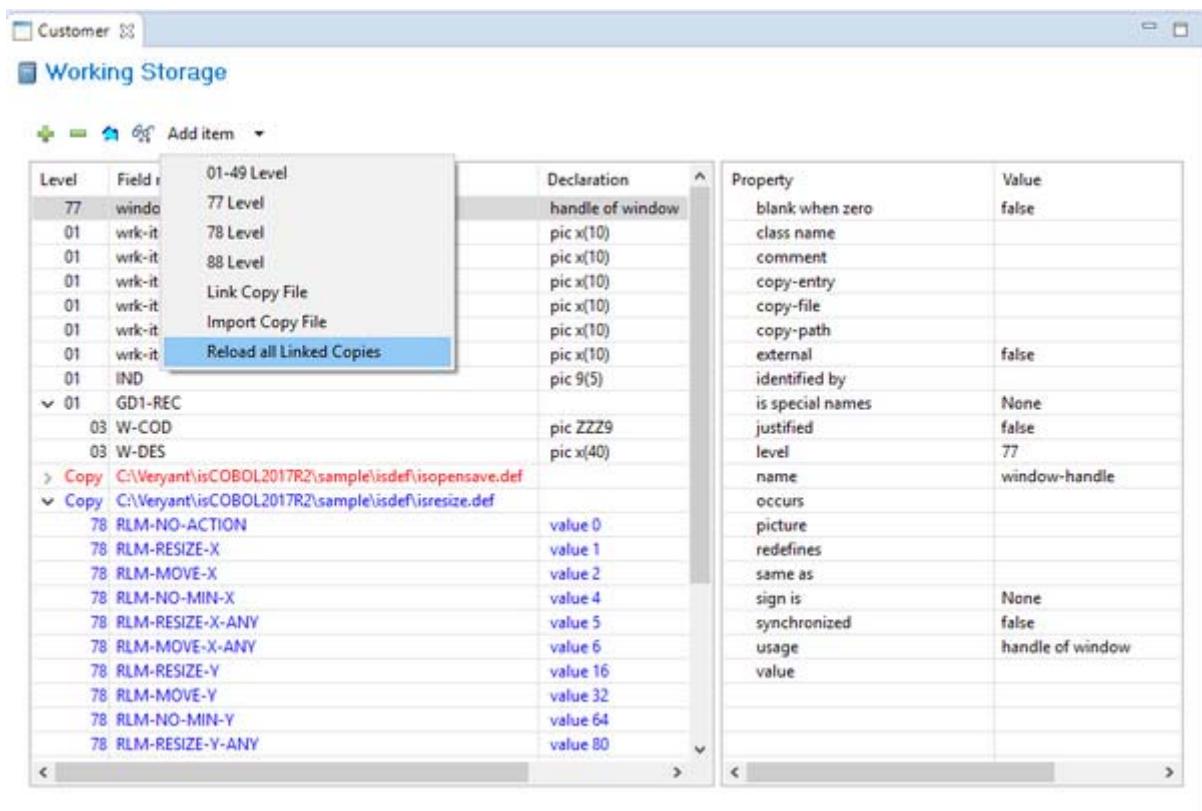
When creating a new screen or report, a template can be selected and a prefix can be specified to generate unique control names.



The prefix can also be changed at a later time, using the Change Prefix dialog.



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.



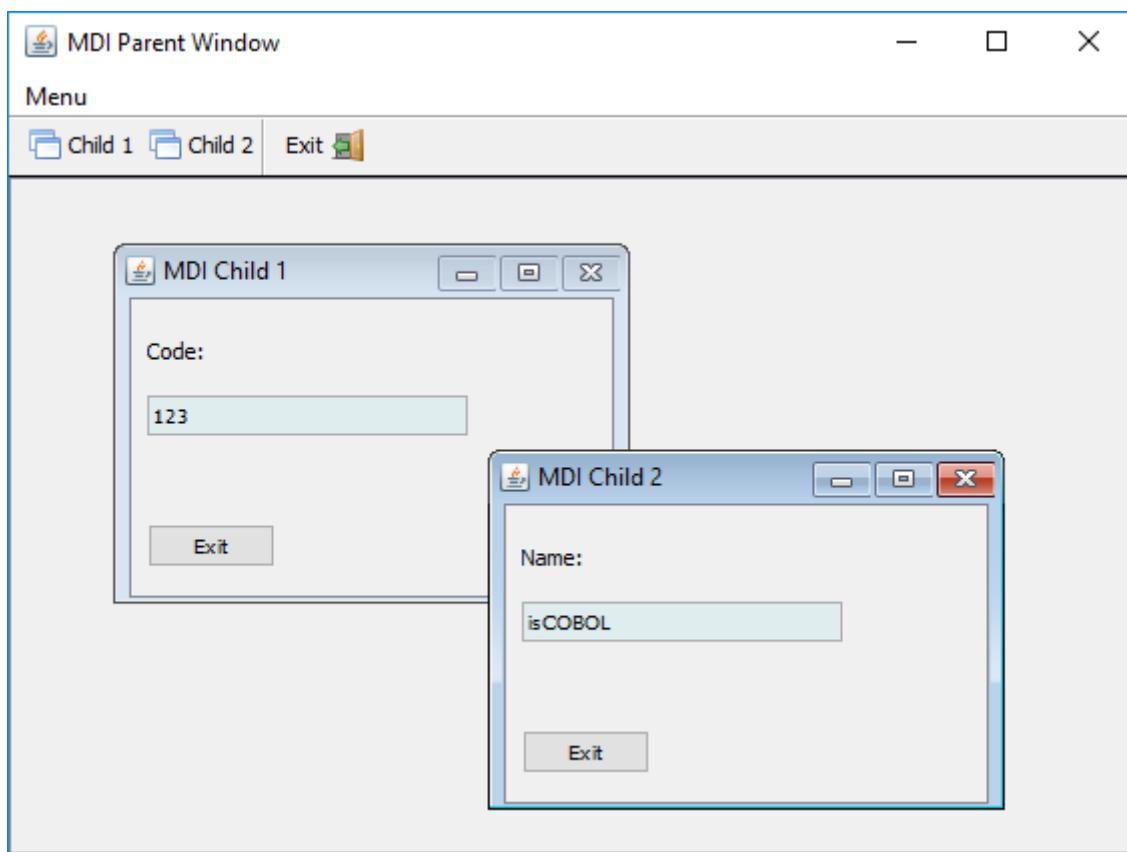
## New User Interface Features

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.



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:

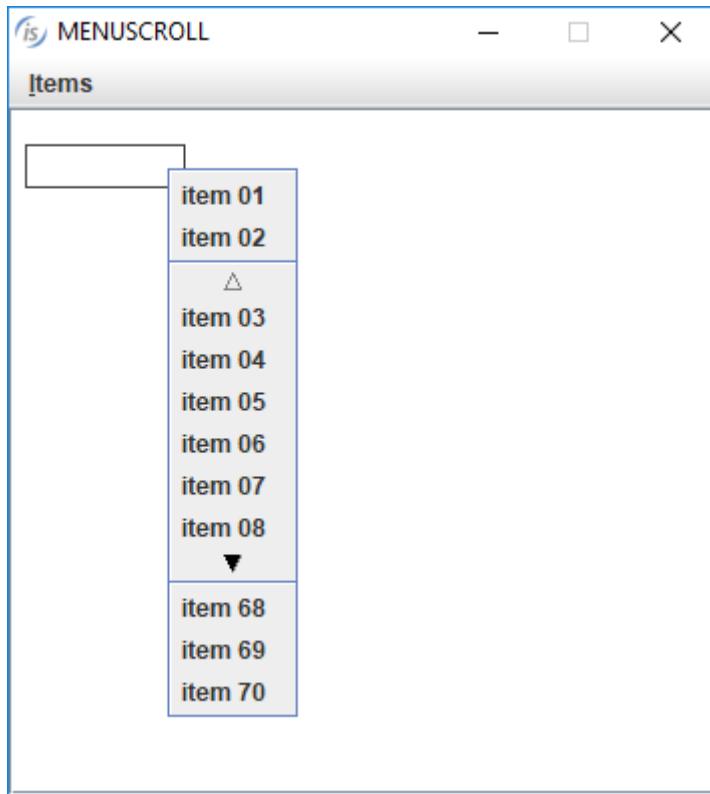
- ScrollItems is the number of visible items that can be scrolled
- FixedTopItems is the number of fixed and always visible items at the top of the menu
- FixedBottomItems is the number of fixed and always visible items at the bottom of the menu

- 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



## 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.

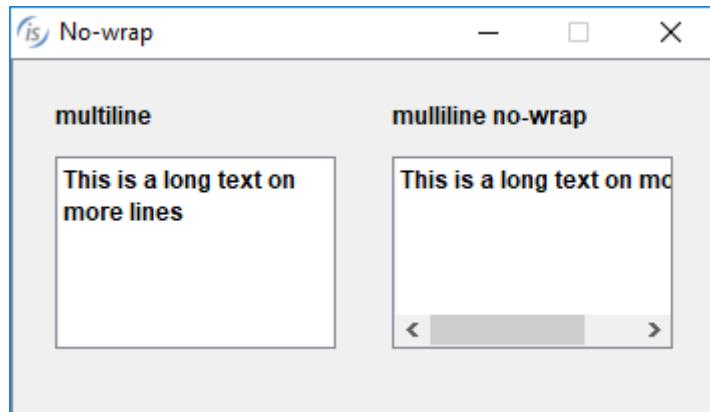
CS-FLAG (numeric) is an optional parameter; when set to 1 the operation is executed on the client, otherwise it is executed on the server.

Code Snippet:

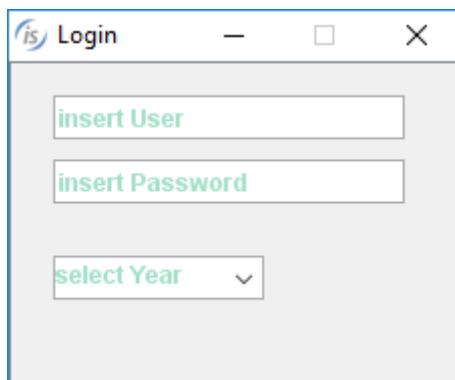
```
move "c:\tmp\myfile.png" to w-uri  
call "C$DESKTOP" using cdesktop-open, w-uri
```

## Other enhancements

A new NO-WRAP style has been added to multiline entry-fields to avoid text wrapping on the next line.



The new configuration option iscobel.gui.placeholder\_color is provided to set the color used for placeholder in entry fields.



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 combobox 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 running on an isCOBOL Server architecture to copy to and from clients.

The table below compares file transfer performance between isCOBOL 2017R2 and the previous version.

Type of CALL "C\$COPY"	File size of file	isCOBOL 2017 R1	isCOBOL 2017 R2
copying a Binaryfile 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 Binaryfile 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. Default 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. The default value for this property is false.

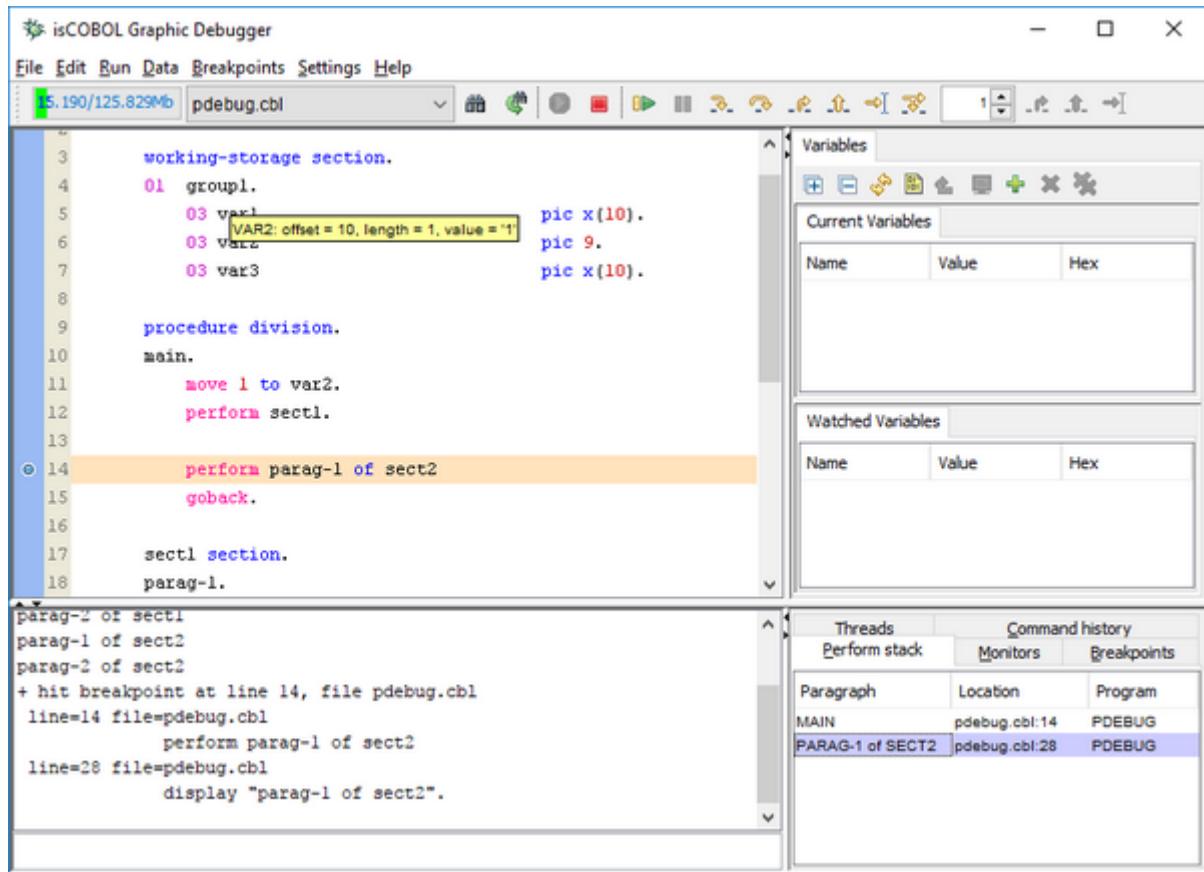
## isCOBOL Compiler and Debugger Enhancements

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.

The figure below 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.



## 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 ip1and 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.

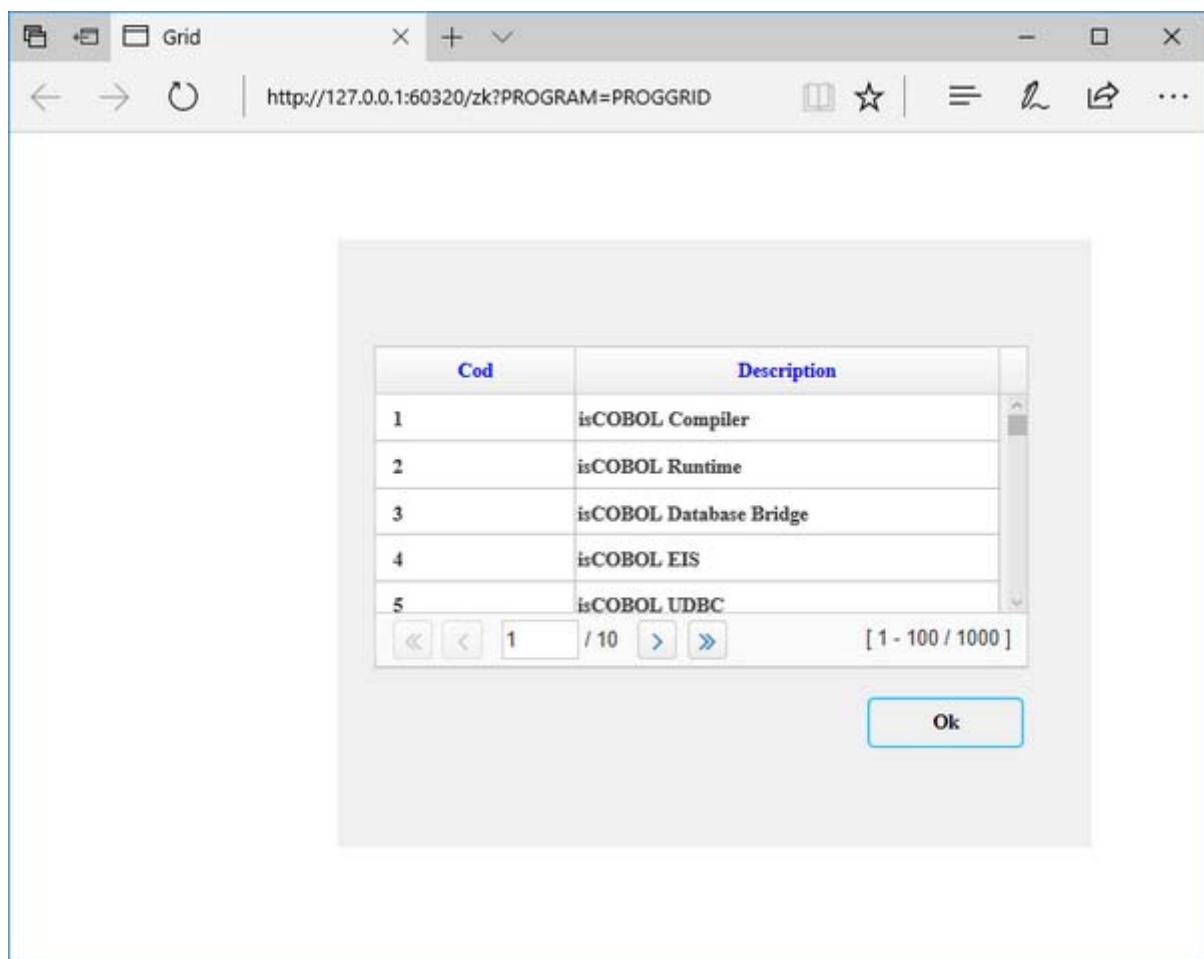
The new property ROWS-PER-PAGE has been added to the Grid component, to set the number of visible rows in the grid and to enable pagination. The COBOL program can load all the available data on the grid, and by setting ROWS-PER-PAGE to a number greater than zero, the runtime will activate automatic pagination.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

```
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
```

The figure below shows the ouput on video.



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 webDirect 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');"
```

# isCOBOL 2017 Release 1 Overview

## Introduction

Veryant is pleased to announce the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2017 R1.

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

The isCOBOL IDE is now based on Eclipse 4.5 Mars release, bringing powerful new aids to the developers, such as a Terminal View for command line use, GIT integration and filtered views.

List-box control has been revamped, by adding check boxes and radio buttons for easy multi or single item selection. Contents in list-box can now be exported to Microsoft Excel XLS and XLSX formats or copied to the system clipboard.

isCOBOL applications can now create and handle multi-platform system tray icons, which developers can leverage to bring even more features to end users.

Entry fields can now be spell-checked automatically, and allow placing bitmaps in both left and right edges of the text box to provide new user interface capabilities.

New library routines ease the job of user interface enhancements with easy to use calls.

The framework has been updated, with the new CrreeJ interface with high-performance in heady multi-threaded environments.

A new IsSort utility allows command-line or programmatic sorting, merging and filtering of indexed, relative and sequential files.

The compiler now creates more optimized classes, that use less memory and load faster, and can generate warning for unsupported statements in isCOBOL EIS html based solutions, and provide better migration support when moving from other COBOLs.

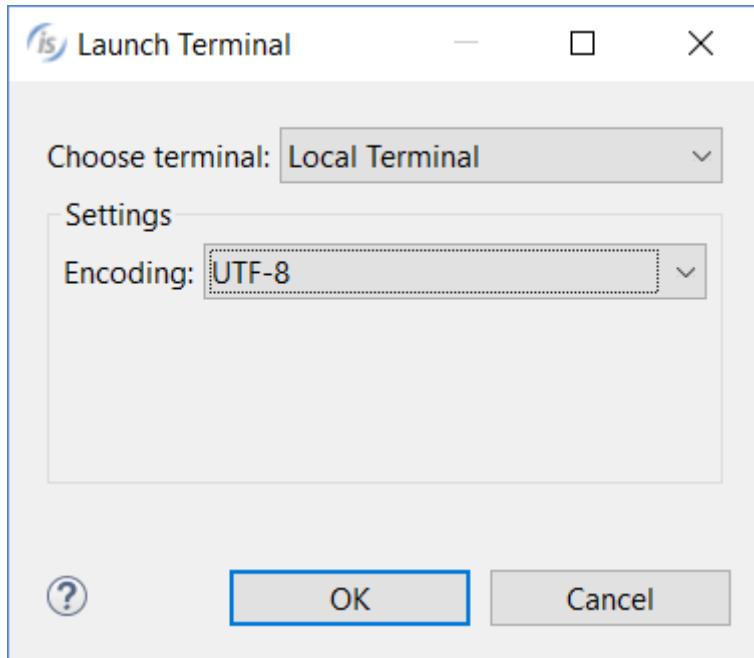
Application Server can now spawn multiple JVM processes and Thin Client applications consume less bandwidth for common statements.

Details on these enhancements and updates are included below.

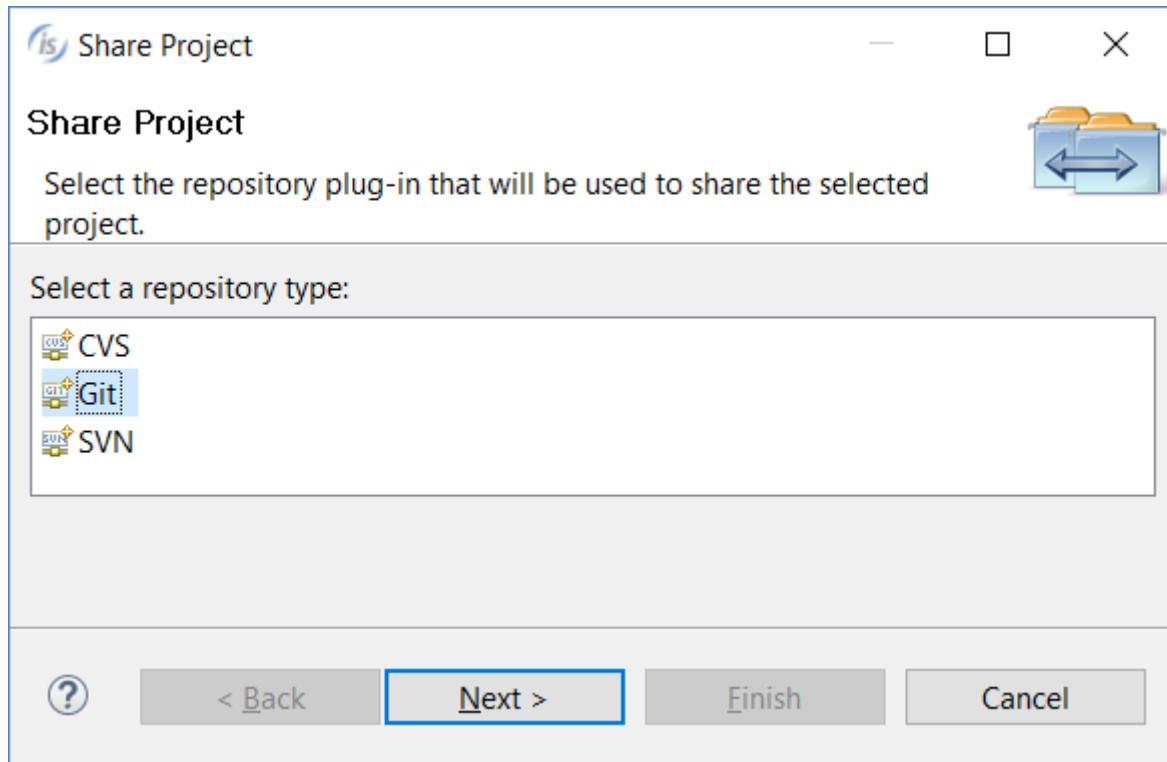
## isCOBOL IDE Enhancements

The isCOBOL 2017R1 IDE is now based on Eclipse 4.5 Mars. These are some of the new features introduced with the new isCOBOL IDE:

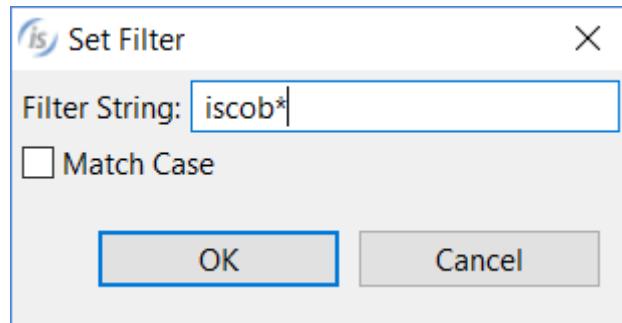
- New IDE dark theme, which looks great for dark lovers.
- Powerful terminal emulator, providing access to the system terminal directly from the IDE, as shown in Figure 1, Terminal configuration

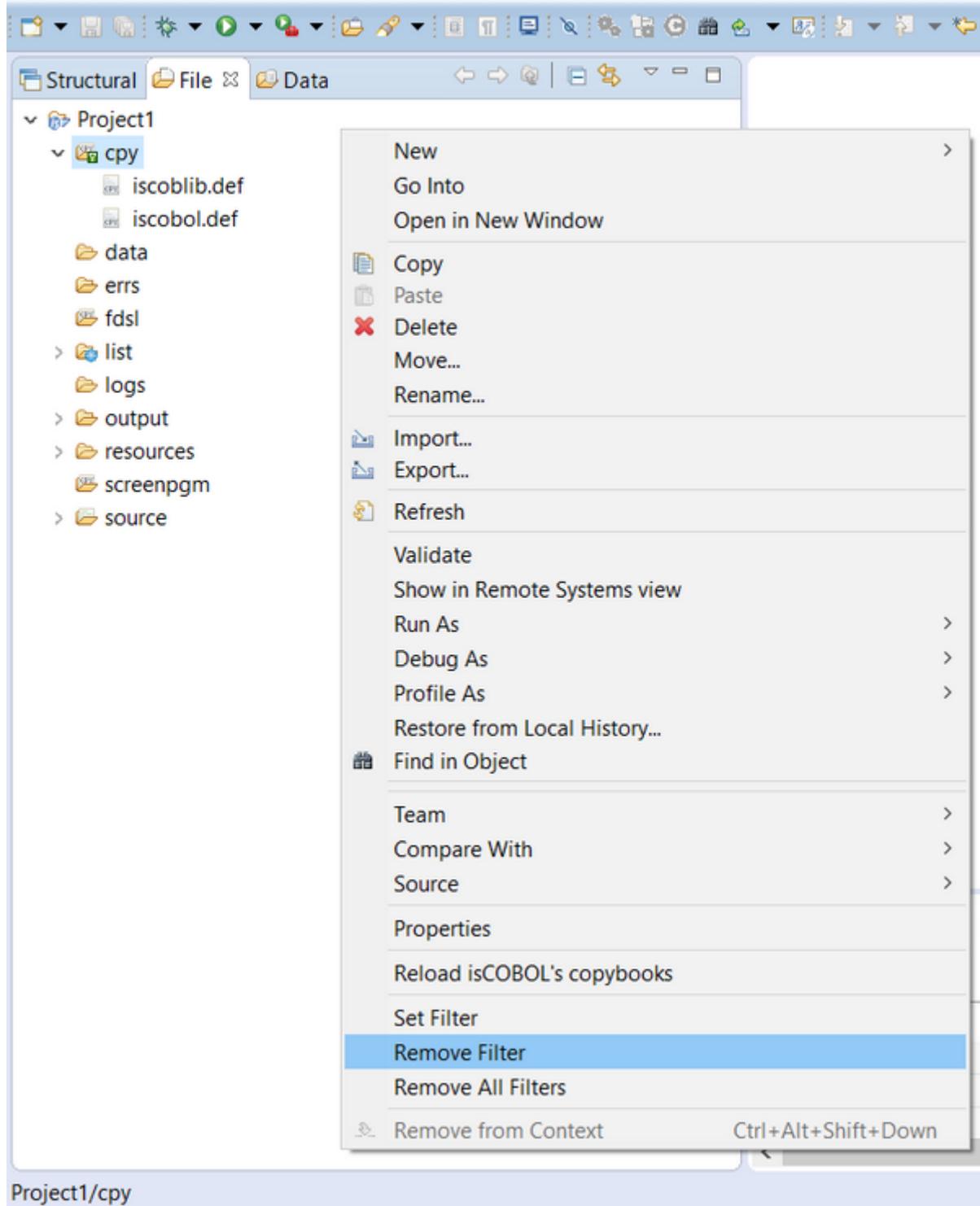


- New keyboard shortcuts to split editors horizontally ( CTRL+\_) or vertically ( CTRL+{ ) allowing editing of two parts of a file at the same time.
- Native support of Git flow



- New meta-character filtering in the structural, file, and data view for easy navigation of large projects, as shown in the next pictures:

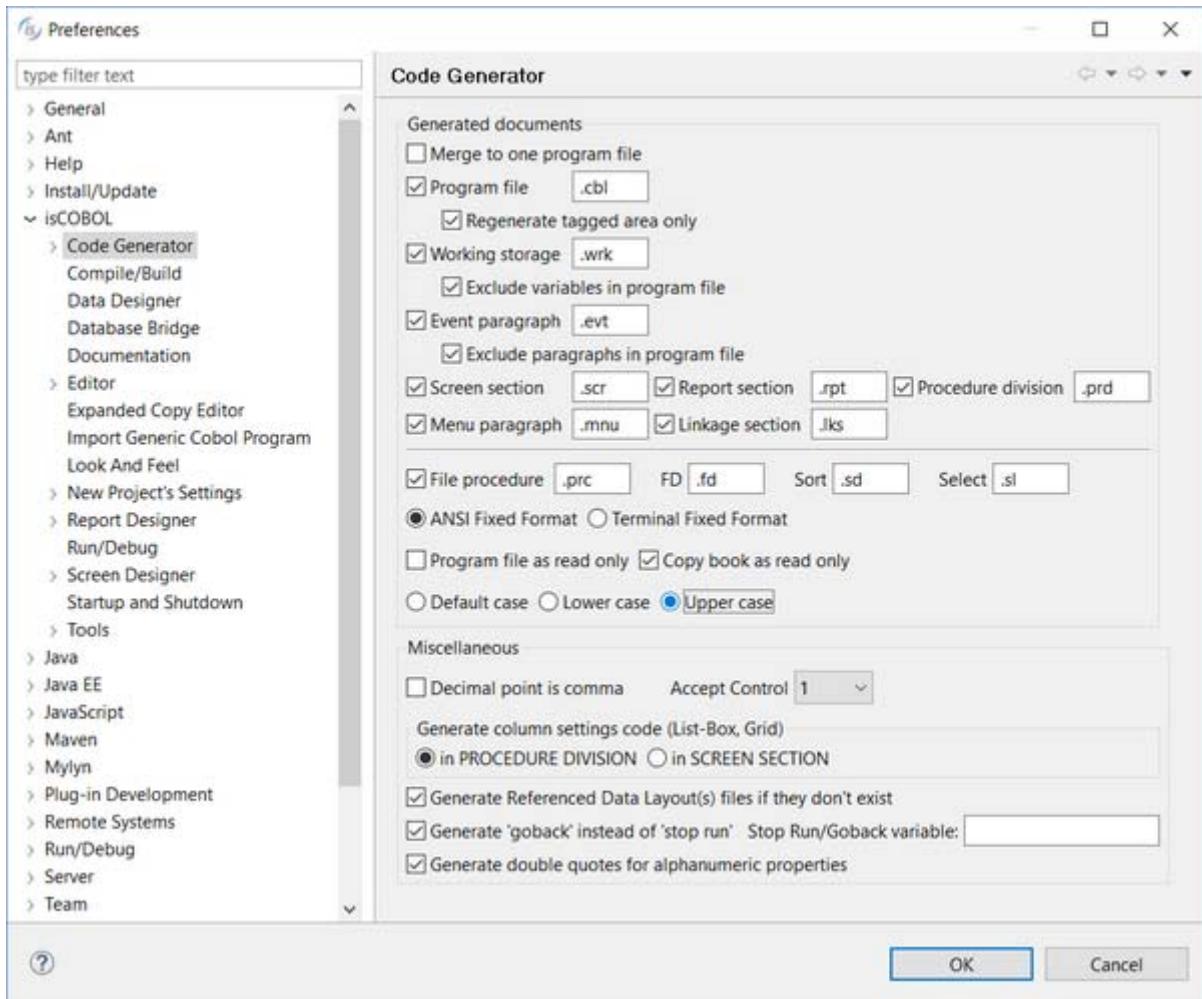




Eclipse's Local history feature is extended to support screen and report painters (.isp files) and file designer (.idl files). This allows easy rollback of recent code changes without using additional source versioning software.

Moreover, .isp files representing Screen Programs and .idl files representing a File Descriptions can be linked into the project, using Eclipse's file linking capabilities, allowing sharing of the source code between projects and developers.

Additional options on code generation have been introduced, to allow excluding automatically generated variables or paragraph, if they are already defined in the source code manually maintained outside the tagged areas. File extensions in the file procedure sections can now be customized, and code can be generated in upper or lower case.



Extensive enhancements have been made in the AcuBench migration wizard, improving compatibility with AcuBench 10, the ACUCOBOL-GT IDE.

## New User Interface Features

The Entry-field control has been greatly enhanced with new features, and List-box has been upgraded as well. Also new is the ability to create a system tray icon.

New library routines have been implemented to simplify and enhance User Interface handling.

## Entry-field control

Entry-fields now support bitmaps inside the entry-field. This allows showing a bitmap on either the left or right edge of the control, or on both. Click and double-click events can be intercepted, to programmatically implement behaviors when the user interacts with these bitmaps. A specific hint message can be set for each bitmap.

This is the list of all new properties available for this feature:

- BITMAP-HANDLE to specify the handle of the loaded bitmap
- BITMAP-WIDTH to set the width of each bitmap when a strip of bitmaps is used
- BITMAP-NUMBER to set the bitmap number to be displayed on the left edge
- BITMAP-DISABLED to set the bitmap number to be displayed on the left edge when the control is disabled
- BITMAP-ROLLOVER to set the bitmap number to be displayed on the left edge when the mouse is over the bitmap
- BITMAP-TRAILING-NUMBER to set the bitmap number to be displayed on the right edge
- BITMAP-TRAILING-DISABLED to set the bitmap number to be displayed on the right edge when the control is disabled
- BITMAP-TRAILING-ROLLOVER to set the bitmap number to be displayed on the right edge, when the mouse is over the bitmap
- BITMAP-HINT to set the hint text of the bitmap on the left edge
- BITMAP-TRAILING-HINT to set the bitmap on the right edge

The following new events are available on entry-field controls:

- MSG-BITMAP-CLICKED is fired when the image is single clicked
- MSG-BITMAP-DBLCLICK is fired when the image is double clicked

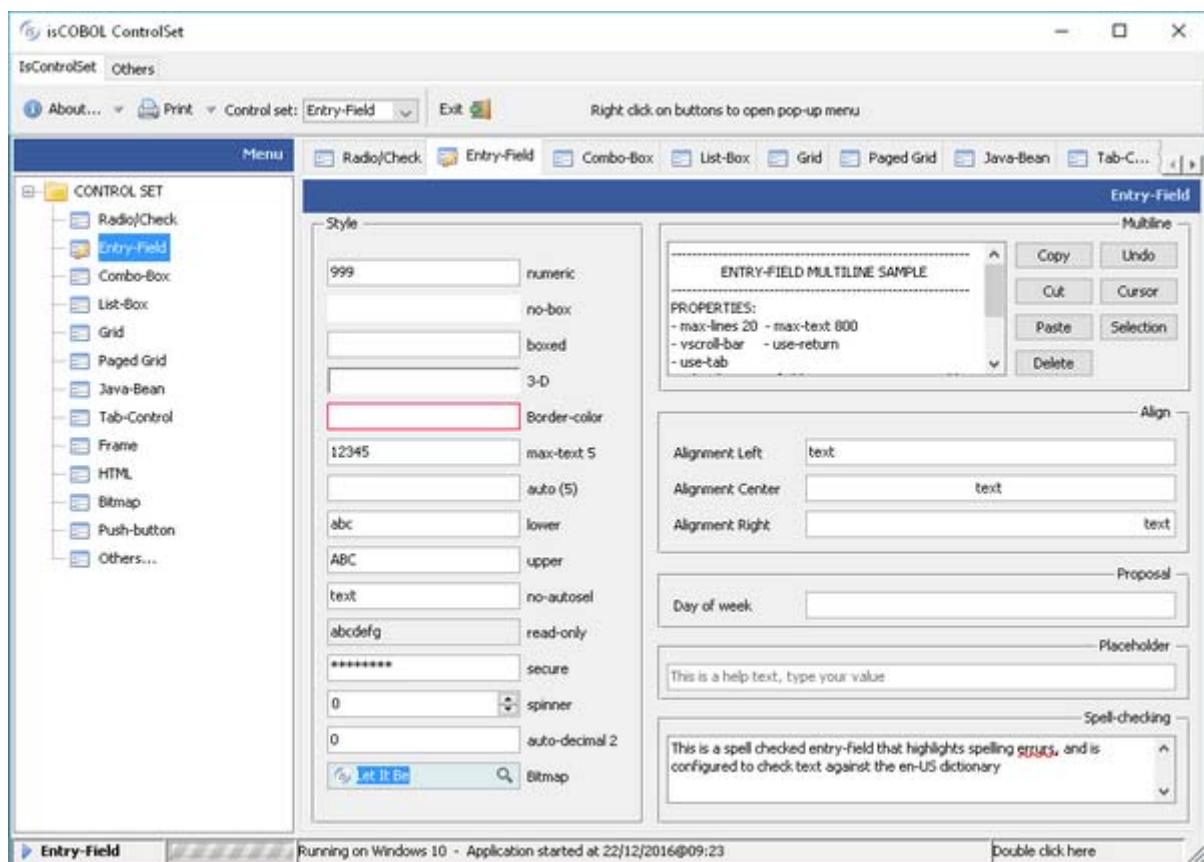
Entry-field controls now also support a new Spellchecker feature with suggestions, using the new property SPELL-CHECKING that can be set to a dictionary language.

The following code is used in the provided ISCONTROLSET sample, where the bitmaps and spell-checking features are used in the entry-fields shown in the last line:

```
05 ef-bmp entry-field
      bitmap-handle h-ef-icon
      bitmap-width 16
      bitmap-number 1
      bitmap-trailing-number 3
      bitmap-trailing-rollover 2
      bitmap-hint "Click here to copy text to clipboard"
      bitmap-trailing-hint "Click here to select a song"
      event EF-BMP-EV

05 spellcheck entry-field
      spell-checking "en-US"

EF-BMP-EV.
evaluate event-type
when msg-bitmap-clicked
  evaluate EVENT-DATA-1
  when 1
    modify ef-bmp cursor -1
    modify ef-bmp action action-copy
  when 2
    set event-action to event-action-terminate
    set perform-lookup to true
  end-evaluate
end-evaluate.
```



## List-box control

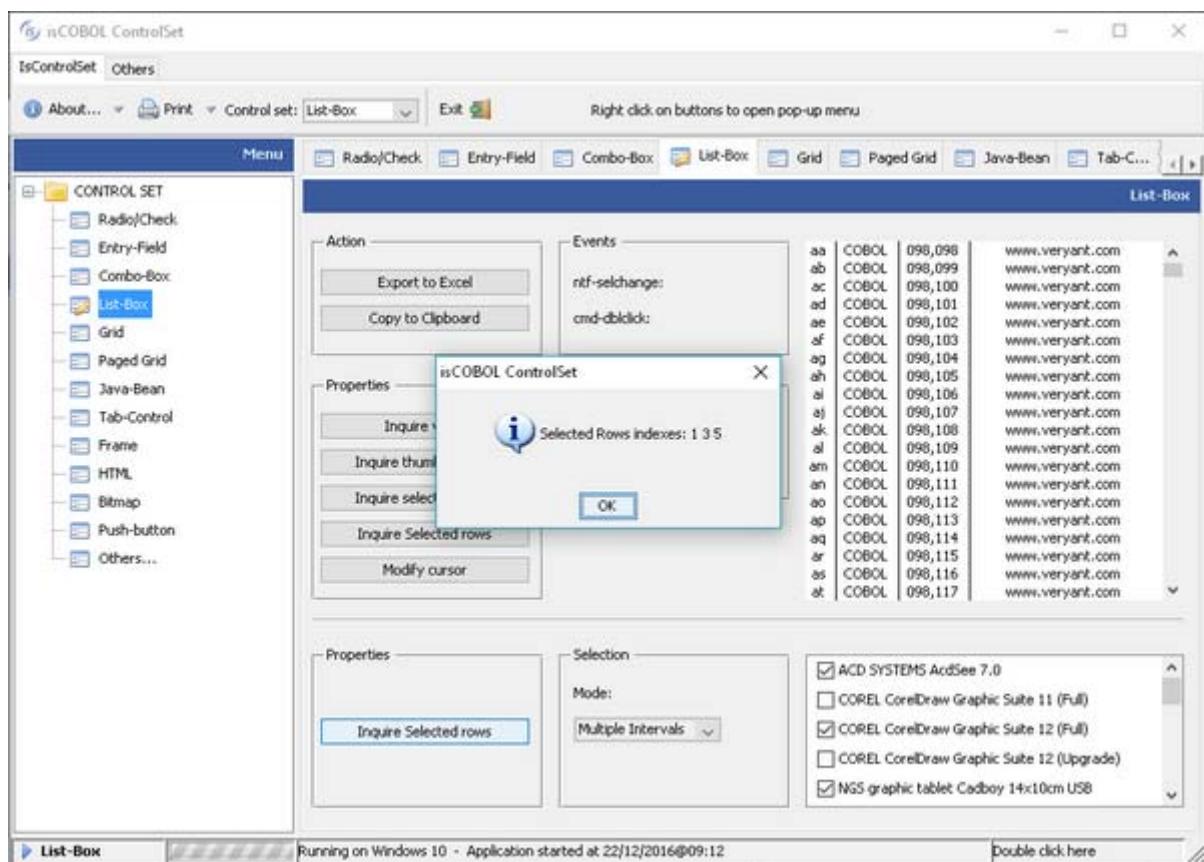
List-box now supports easy multi-selection with check-boxes, as shown in Figure 7, Check-list multiple selection, or single-selection with radio buttons. Those features can be enabled by setting the new style CHECK-LIST and property SELECTION-MODE to specify if a single or multiple selection is to be used. Selected items can be inquired with the property ROW-SELECTED.

Below is a code sample demonstrating the use of CHECK-LIST style and SELECTION-MODE property of list-box:

```

05 my-list-multiple list-box
      check-list
      selection-mode lssm-multiple-interval-selection
05 my-list-single list-box
      check-list
      selection-mode lssm-single-selection
      inquire my-list-single  rows-selected w-row
      inquire my-list-multiple rows-selected w-rows

```



List-box content can now be copied to the clipboard and exported in Microsoft Excel XLS and XLSX formats. The new features can be added automatically or controlled by code.

Using COBOL code, the ACTION property value ACTION-EXPORT will trigger the list-box data export feature. Exported data file name and format can be customized using the EXPORT-FILE-NAME and EXPORT-FILE-FORMAT properties.

Using the ACTION-COPY value of the ACTION property will copy the list-box contents to the clipboard.

Multiple selection modes are now also supported in the list-box control, to allow users to more conveniently select items.

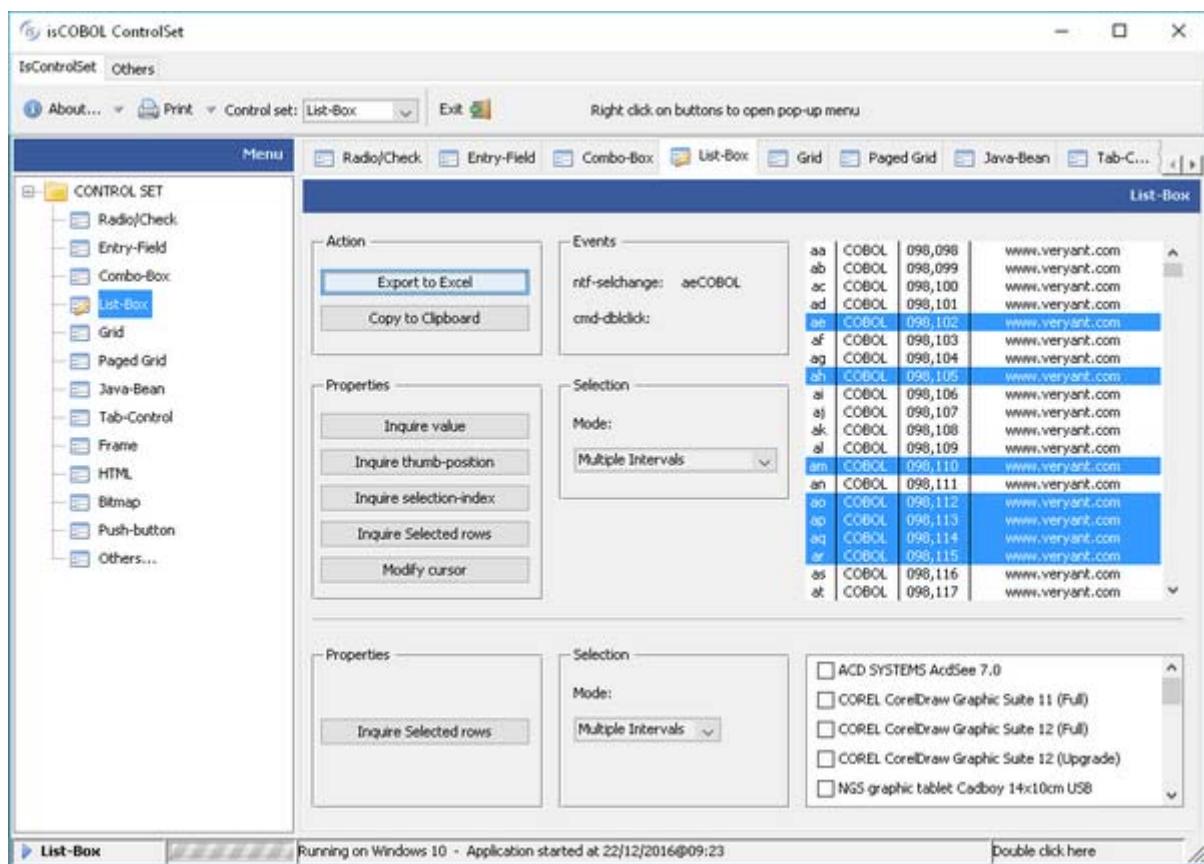
#### New properties in the LIST-BOX control

- SELECTION-MODE to set the selection type
- ROWS-SELECTED to retrieve the selected items list

The next picture shows how the user can access the new list-box export feature. This can be achieved by properly setting the export properties of the list-box, as shown below.:

```
05 my-list list-box
      selection-mode lssm-multiple-interval-selection
      export-file-name w-path-filename
      export-file-format "xlsx"

      modify my-list action action-export
```



The data exported to Excel will look like this

A	B	C	D	E	F	G	H
1 aa	COBOL	098,098	www.veryant.com				
2 ab	COBOL	098,099	www.veryant.com				
3 ac	COBOL	098,100	www.veryant.com				
4 ad	COBOL	098,101	www.veryant.com				
5 ae	COBOL	098,102	www.veryant.com				
6 af	COBOL	098,103	www.veryant.com				
7 ag	COBOL	098,104	www.veryant.com				
8 ah	COBOL	098,105	www.veryant.com				
9 ai	COBOL	098,106	www.veryant.com				
10 aj	COBOL	098,107	www.veryant.com				
11 ak	COBOL	098,108	www.veryant.com				
12 al	COBOL	098,109	www.veryant.com				
13 am	COBOL	098,110	www.veryant.com				
14 an	COBOL	098,111	www.veryant.com				
15 ao	COBOL	098,112	www.veryant.com				
16 ap	COBOL	098,113	www.veryant.com				
17 aq	COBOL	098,114	www.veryant.com				
18 ar	COBOL	098,115	www.veryant.com				
19 as	COBOL	098,116	www.veryant.com				
20 at	COBOL	098,117	www.veryant.com				
21 au	COBOL	098,118	www.veryant.com				
22 av	COBOL	098,119	www.veryant.com				
23 aw	COBOL	098,120	www.veryant.com				
24 ax	COBOL	098,121	www.veryant.com				
25 ay	COBOL	098,122	www.veryant.com				
26 az	COBOL	098,123	www.veryant.com				

## Tray icon

In W\$MENU a new op-code named WMENUNEW-TRAY has been implemented to manage a multi-platform system tray icon.

The menu is shown in the system tray as an icon where the user can either left or right click or double click. With a right click, the sub-menu (if any) is shown, while with a left and double click, exceptions are sent to the current ACCEPT

When calling w\$menu with op-code wmenu-new-tray, the text passed in the second parameter becomes the hint/tooltip shown when the user hovers the mouse over the tray icon, the third parameter is the exception value generated when the user clicks the icon, and the fourth parameter is the exception value generated when double clicking. The next three parameters are used to select the bitmap to be displayed from the bitmap strip.

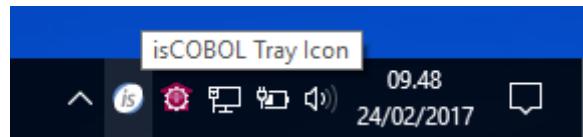
Using the calling w\$menu with op-code wmenu-add menu items can be added to the tray icon, as in a normal menu-bar.

The following is an example of the new tray icon usage.

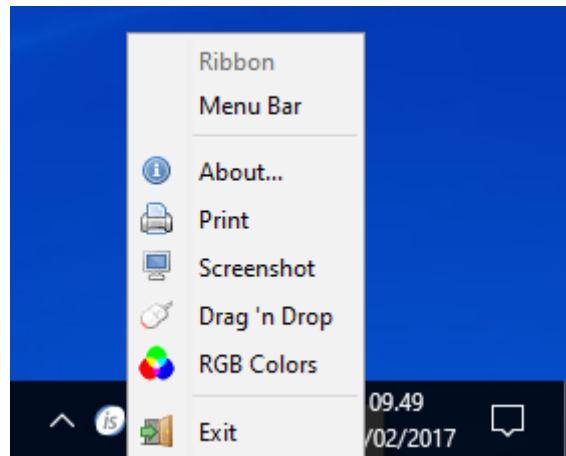
```
call "w$menu" using wmenu-new-tray
      "isCOBOL Tray Icon"
      78-tray-click-exc
      78-tray-double-click-exc
      h-tray-icon 1 16
      giving h-tray-menu.

call "W$MENU" using wmenu-add
      h-tray-menu
      0 0
      "&About . . ."
      menu-pb-about
```

Running the code above will display the tray icon



Right clicking on the icon shows a pop-up menu



## Library routines

New routines have been implemented to enhance and simplify User Interface handling.

This is the list of new routines:

- W\$SAVE-IMAGE saves a bitmap-handle to a disk file
- W\$HINT shows hints programmatically. Hints can also have a different hide-timeout than the default one set from configuration
- W\$PROGRESSDIALOG easily shows a progress dialog with determinate or indeterminate percentage indicator
- W\$CENTER\_WINDOW centers a window in the screen without needing any calculation from code

Enhancements in existing routines:

The new op-code WBITMAP-LOAD-FROM-CLIENT in W\$BITMAP routine has been added to manage client-side bitmaps in Thin Client architecture. This is also useful to increase Thin Client performance.

## Framework improvements

isCOBOL Evolve 2017 R1 includes a new interface to access C-Tree files and a new utility named IsSort that allow to sort, merge and filter indexed, relative and sequential files.

### C-Tree new interface

A new c-treeRTG interface, CtreeJ, is now qualified to be used in heavy multi-thread environments, improving performance of Client/Server architectures of up to 50% on read statements and up to 20% in update statements when compared to other interfaces for c-treeRTG.

CtreeJ fully supports c-treeRTG bound server architecture.

The new interface can be enabled modifying a setting the isCOBOL configuration file:

```
iscobol.file.index=ctreej
```

When using Bound server, also include:

```
iscobol.ctree.bound_server=true
```

Previous values, "fscsc" for connector and "ctree2" for native library, are still supported for backward compatibility.

### IsSort utility

The new command line utility IsSort and the new C\$SORT routine can be used to sort, merge and filter indexed, relative and sequential files.

The utility is available from the command line or through a COBOL CALL statement

Parameters can be used to drive the new feature:

- SORT/MERGE to specify either a sort or a merge option
- FIELDS to specify the field names of the file to be used
- USE/GIVE to allow to specify the input and output files, of the desired process
- INCLUDE/OMIT inclusion or omission of selected records

When using the command below in a terminal window

```
issort take sort.cmd
```

and the file sort.cmd contains:

```
sort fields (1, 6, ch, d)
use idxfile org ix
record f 40
key (1, 6, p, 7, 15, c, 22, 15, ad)
give output.txt org ls
record f 40
include cond = 37,4,ge,902
```

IsSort reads the indexed file called idxfile, with record size of 40 bytes, primary key made by two segments and containing an alternate key which allows duplicates.

Records are sorted in descending order on the field with offset=1 and size=6 (the first primary key segment)

Records in which the field at offset 34 with size 4 bytes don't contain a number equal of greater than 902 are ignored, and will not be saved in the output file.

Sorted records are saved to a line sequential file named output.txt.

All parameters needed for the sort operations can be specified directly on the command line, as shown below

```
issort sort fields (1, 6, ch, d) use idxfile org ix record f 40key (1, 6, p, 7, 15, , 22, 15, ad) give output.txt org ls record f 40 include cond = 37,4,ge,902
```

Sorting can be invoked programmatically using a call statement:

```
call "c$sort" using "sort fields (1, 6, ch, d) use idxfile org ix record f 40 key (1, 6, p, 7, 15, , 22, 15, ad) give output.txt org ls record f 40 include cond = 37,4,ge,902"
```

## isCOBOL Compiler Enhancements

isCOBOL Evolve 2017 R1 includes several changes to the isCOBOL Compiler that improve productivity and simplify migration from other COBOLs.

### Compiler optimizations

Recompiling programs with 2017 R1 creates more optimized classes that use less memory and provide faster class-loading times. This optimization is enabled by default, no options required.

An additional compiler option, -oe, has been added to optimize the java code for the EVALUATE statement with string literals. This takes advantage of the Java statement "switch" on Strings, supported by JDK 1.7.

### Compiler warning detection

A new compiler option, -whttp, is now available to show Warnings for unsupported statements under isCOBOL EIS html based solution, useful when developing isCOBOL Mobile applications. This helps in migrating existing COBOL programs to web or Mobile Apps.

### Enhanced compatibility with other COBOLs

The WAIT LOCK clause in READ statements is now supported by the compiler, and is currently implemented in Jlsam and C-tree interfaces, to fully support the Micro Focus COBOL syntax.

A new compiler option, -crlk, has been implemented to emulate the RM-COBOL style lock modes, whose behavior depends on DECLARATIVES for specific files.

A new configuration property,

```
iscobol.ccopy.client_temp_as_base_dir=true
```

has been added, which will use the client's temp folder in C\$COPY when copying files in Thin Client architecture. This enhances compatibility with ACUCOBOL-GT.

## IsCOBOL Server Improvements

isCOBOL Server can now be configured to start additional JVM processes on the server side when receiving connection requests. Moreover, the administrator Panel has been enhanced.

Thin Client TCP/IP traffic generated on statements such as:

- `display window`
- `inquire line, col, size, lines`
- `call "w$font"`

has been reduced up to 50% enhancing performance.

### Multitasking

This new feature helps when multiple developers need to debug applications running on the same isCOBOL Server.

The new configuration setting, `iscobol.as.multitasking` allows specifying whether separate JVM processes are used for every thin client (value 1) or only the client programs launched in debug mode (value 2).

Java options can now be passed to the new processes by setting the new configuration `iscobol.jvm_options`.

Following is a configuration sample used to configure isCOBOL Server for debugging simultaneously from 2 different Clients:

```
iscobol.as.multitasking=2
```

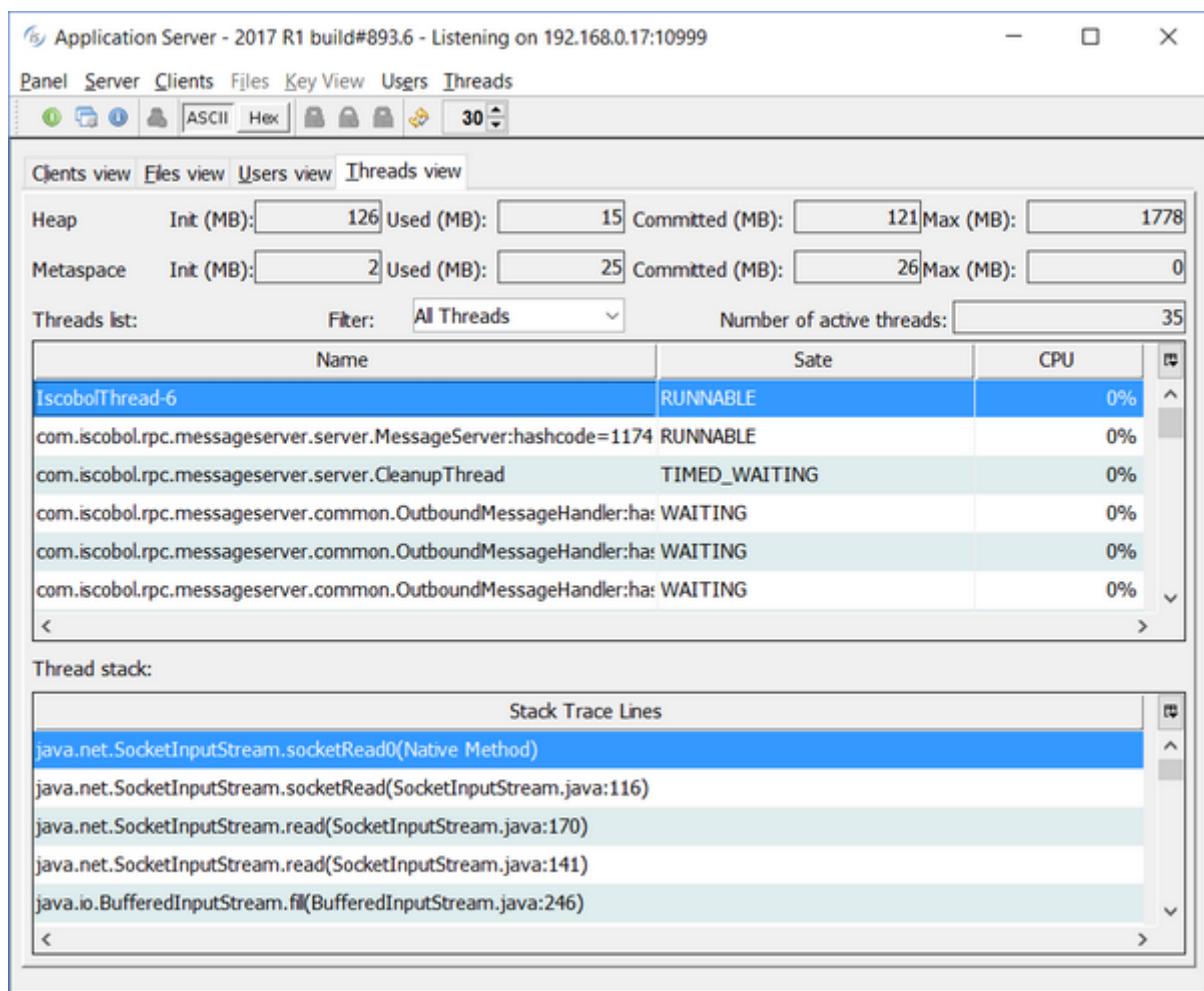
Using this setting two developers can debug Thin Client applications by only selecting different debug ports:

```
iscclient -hostname ipserver -port 10999 -d -debugport 9991 MYPROG  
iscclient -hostname ipserver -port 10999 -d -debugport 9992 MYPROG
```

These debugger sessions don't interfere with other Clients running without debug mode, allowing debugging applications on production or test environments, while other clients are running.

### Panel improvements

The isCOBOL Server administration Panel has a new "Threads View", which shows CPU usage of all isCOBOL Server threads:



The Clients View in the Panel now has the new Stack column, that can now show the list of running threads for each TID, if the application uses multithread programming, and the COBOL stack trace for each thread

Application Server - 2017 R1 build#893.6 - Listening on 192.168.0.17:10999

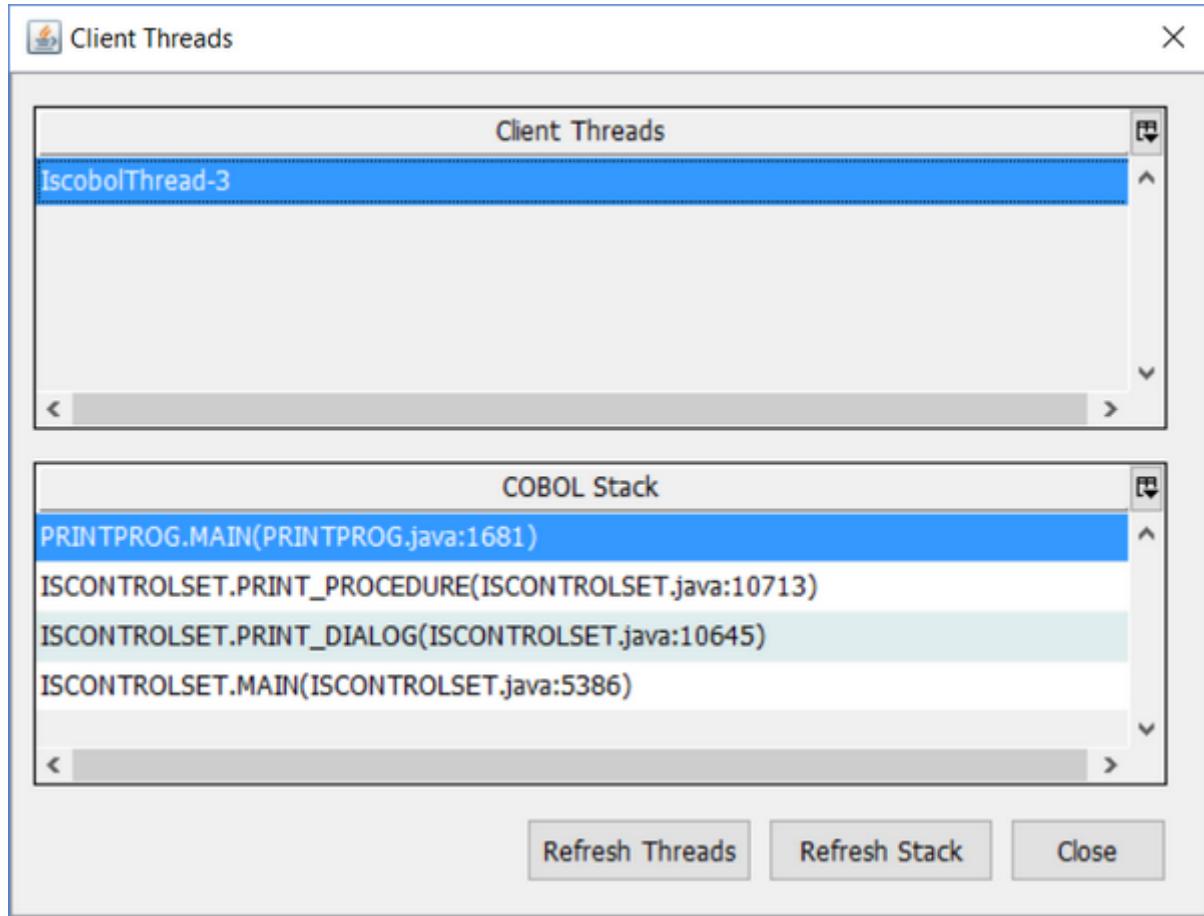
Panel Server Clients Files Key View Users Threads

Clients view Files view Users view Threads view

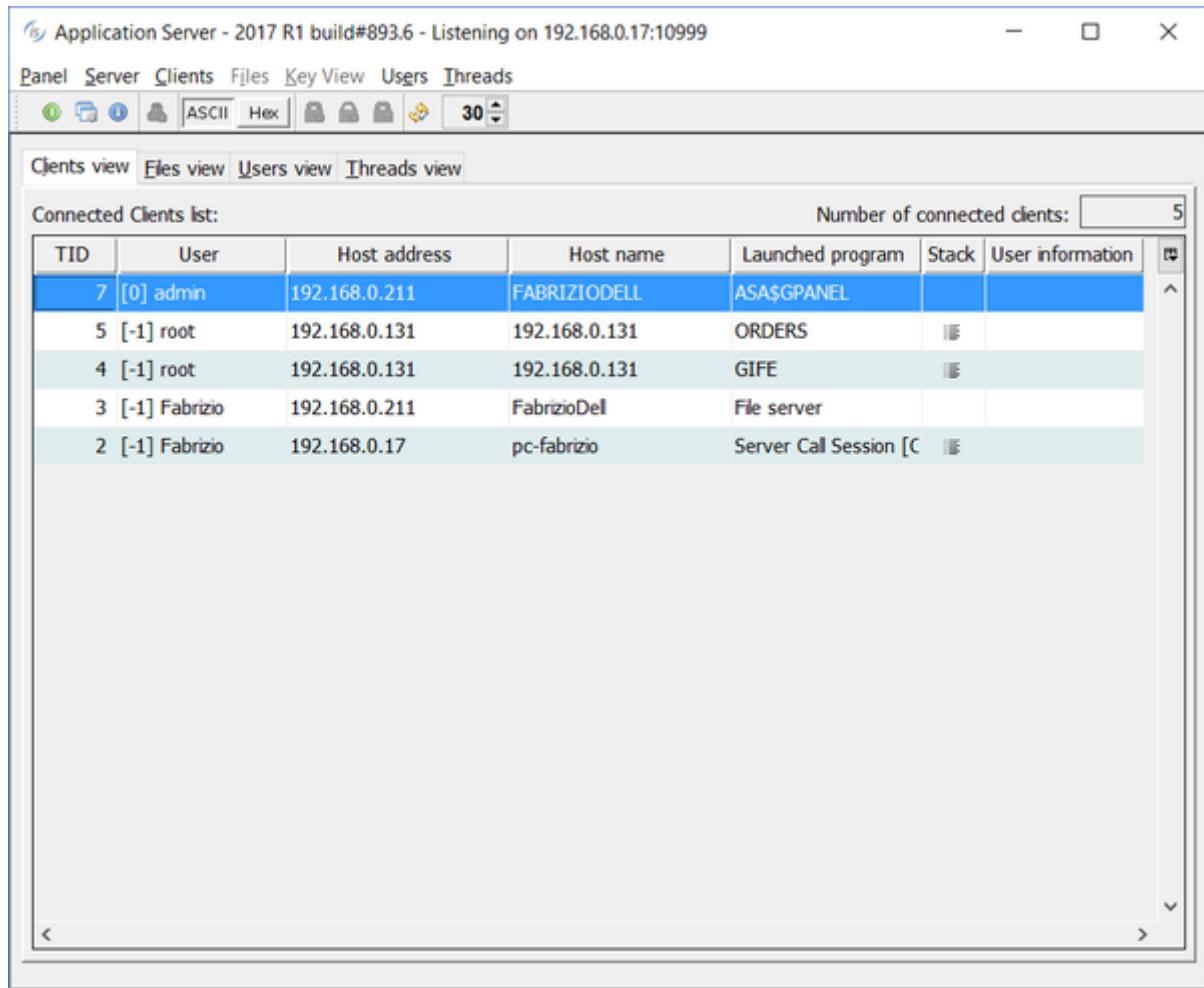
Connected Clients list: Number of connected clients: 5

TID	User	Host address	Host name	Launched program	Stack	User information
7	[0] admin	192.168.0.211	FABRIZIODELL	ASA\$GPANEL		
5	[-1] root	192.168.0.131	192.168.0.131	ORDERS		
4	[-1] root	192.168.0.131	192.168.0.131	GIFE		
3	[-1] Fabrizio	192.168.0.211	FabrizioDel	File server		
2	[-1] Fabrizio	192.168.0.17	pc-fabrizio	Server Call Session [C]		

The stack trace of a single thread program looks like this



The stack trace of a multi thread program looks like this



## isCOBOL EIS improvements

### COBOL Java-Bean client generation

The compiler can now generate clients for web services (REST or SOAP) that can be used in a COBOL or Java context. While generating the service bridge program, the compiler can now also generate a client Bean for easy web service testing, for example is a JavaServer Page (JSP).

For example, if the Service Bridge is used to generate a Rest web service, the bean must be configured as

```
iscobol.compiler.servicebridge.bean=rest
```

while if a SOAP web service was generated, then the configuration should be

```
iscobol.compiler.servicebridge.bean=soap
```

Other configuration options can be set to further customize code generation

```
iscobol.compiler.servicebridge.bean.prefix=prefix (default bean)
iscobol.compiler.servicebridge.bean.url=shttp://myip:8081/myservices (default "http://
localhost:8080/services")
```

## Log http requests and responses

Http requests and responses handled by the `HTTPHandler` class can now be logged with new configuration options:

```
iscobol.soap.log=true
```

globally enables logging of web service methods. Set

```
iscobol.soap.log.methodname=true
```

to enable or disable logging for the SOAP service with the specified method name, overriding the global setting where needed.

```
iscobol.soap.log.folder=/path
```

sets the folder where .log files are generated.

Logging can be fine-tuned by globally enabling or disabling logging for every implemented method in the web service, and overriding settings for individual methods.

The new log feature aids in debugging SOAP web services by logging the raw HTTP requests received from clients, including HTTP headers and SOAP XML, before they are parsed by the runtime, and the raw SOAP XML responses sent back to the clients.

Following is a sample of the data logged by the new feature:

```
=====
Request received at 2016-12-354 - 11.14.53.0758
Begin request id:1mj2hvpsys17wne8lbwev9wb
Request headers:
User-Agent=Java/1.8.0_65
Connection=keep-alive
Host=127.0.0.1:55171
Accept=text/html, image/gif, image/jpeg, *, q=.2, */*, q=.2
Content-Length=1335
Content-Type=application/soap+xml; charset=utf-8
Request body:
Input Request
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://tempuri.org/SONGS">
    <soapenv:Body>
        <tns:SONGS>
            <tns:lnk_op_code>F</tns:lnk_op_code>
            <tns:lnk_song_data_in>
...
            </tns:lnk_sd_authors_in>
            </tns:lnk_song_data_in>
        </tns:SONGS>
    </soapenv:Body>
</soapenv:Envelope>

Output Response generated at 2016-12-354 - 11.14.53.0771
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:tns="http://tempuri.org/SONGS">
    <soapenv:Body>
        <tns:SONGSResponse>
            <tns:lnk_song_data_out>
                <tns:lnk_sd_id_out>1</tns:lnk_sd_id_out>
                <tns:lnk_sd_title_out>Let It Be</tns:lnk_sd_title_out>
                <tns:lnk_sd_length_out>4:03</tns:lnk_sd_length_out>
                <tns:lnk_sd_artist_out>Beatles</tns:lnk_sd_artist_out>
                <tns:lnk_sd_album_out>Let It Be</tns:lnk_sd_album_out>
                <tns:lnk_sd_genre_out>Pop</tns:lnk_sd_genre_out>
                <tns:lnk_sd_label_out>Apple Records</tns:lnk_sd_label_out>
                <tns:lnk_sd_year_out>1970</tns:lnk_sd_year_out>
                <tns:lnk_sd_authors_out>
                    <tns:lnk_sd_author_out>Paul McCartney</tns:lnk_sd_author_out>
                </tns:lnk_sd_authors_out>
...
                </tns:lnk_song_data_out>
                <tns:lnk_return_status>
                    <tns:lnk_status>OK</tns:lnk_status>
                    <tns:lnk_file_status/>
                    <tns:lnk_status_message>Operation successful</tns:lnk_status_message>
                </tns:lnk_return_status>
            </tns:SONGSResponse>
        </soapenv:Body>
</soapenv:Envelope>
```

## isCOBOL 2016 Release 2 Overview

### Introduction

Veryant is pleased to announce to selected users, the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2016 R2.

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

To allow the “outside world” to interact with an isCOBOL program, isCOBOL 2016 R2 includes a new feature called isCOBOL Service Bridge, integrated in the isCOBOL IDE and also available from the isCOBOL Compiler command line, which allows developers to easily create SOAP and REST Web Services starting from an existing legacy COBOL program.

isCOBOL 2016R2 includes many enhancements and new UI features. For example, grids content can now be effortlessly exported in Microsoft Excel format (both .xls and .xlsx formats).

Details on these enhancements and updates are included below.

### isCOBOL Service Bridge facility

To allow other software to communicate with an isCOBOL program, isCOBOL 2016R2 now provides easy server-side SOAP and REST Web Services development using the isCOBOL Server Bridge feature, available in the EIS framework. With Server Bridge, every time the isCOBOL Compiler compiles a legacy COBOL program with Linkage Section, a bridge class that allows the program to be used as a Web Service is automatically generated.

This feature is enabled by setting the property `iscobol.compiler.servicebridge` to true, and can be customized through the Service Bridge configuration described as follows:

```
iscobol.compiler.servicebridge=true
iscobol.compiler.servicebridge.type=SOAP|REST
iscobol.compiler.servicebridge.package=...
iscobol.compiler.servicebridge.rest.prefix=...
iscobol.compiler.servicebridge.rest.response=JSON|XML
iscobol.compiler.servicebridge.soap.prefix=...
iscobol.compiler.servicebridge.soap.url=...
iscobol.compiler.servicebridge.soap.style=RPC|Document
iscobol.compiler.servicebridge.soap.namespace=...
```

To generate a REST web service with JSON responses, for example, the following configuration should be used when compiling:

```
iscobol.compiler.servicebridge=true  
iscobol.compiler.servicebridge.type=REST  
iscobol.compiler.servicebridge.rest.response=JSON
```

In addition, the generation of the Service Bridge class can be customized using \$ELK directives that need to be set before each data item in Linkage section. For example, using the code sample below, the web service will have an input/output parameter called *code*, an input parameter called *name* and an output parameter called *description*.

When the web service is called, the corresponding linkage data items (*p1*, *p2* and *p3*) will be assigned the corresponding values.

```
Linkage Section.  
01 params.  
$ELK NAME=code  
03 p1 pic 9(9).  
$ELK INPUT, NAME=name  
03 p2 pic x(20).  
$ELK OUTPUT, NAME=description  
03 p3 pic x(100).
```

isCOBOL IDE users can rely on the isCOBOL Service Editor to automatically and graphically generate the needed configuration and directives. Using this editor, developers can map the Linkage Section data items to the Web Service parameters, as well as configure other Web Service specific parameters. As soon as changes are saved, the configuration and original source code is updated with the proper compiler directives. In the isCOBOL Editor you can access the new isCOBOL Service Editor, and graphically customize the Web Service generation.

The screenshot shows a COBOL source code editor window titled "SONGS.cbl". The code is a configuration section for a servicebridge. A context menu is open over the line "PROGRAM-ID. SONGS." at line 19. The menu path "Open With" is highlighted. A secondary submenu is displayed, listing several options:

- isCOBOL Copy View
- isCOBOL Editor
- isCOBOL Service Editor
- Text Editor
- System Editor
- In-Place Editor
- Default Editor
- Other...

The main menu also includes standard file operations like Undo, Revert File, and Save, along with other options like Debug As, Run As, and Preferences.

```
*A'1'B'2'3'4'5'6'7'
1 *> Copyright (c) 2005 - 2016 Veryant. Users of isCOBOL
2 *> may freely modify and redistribute this program.
3 $set "servicebridge" "1"
4 $set "servicebridge.type" "REST"
5 PROGRAM-ID. SONGS.
6 CONFIGURATION SECTION.
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
      01  initial-song-table.
      *** 03 filler pic x(30) value "Let It Be".
```

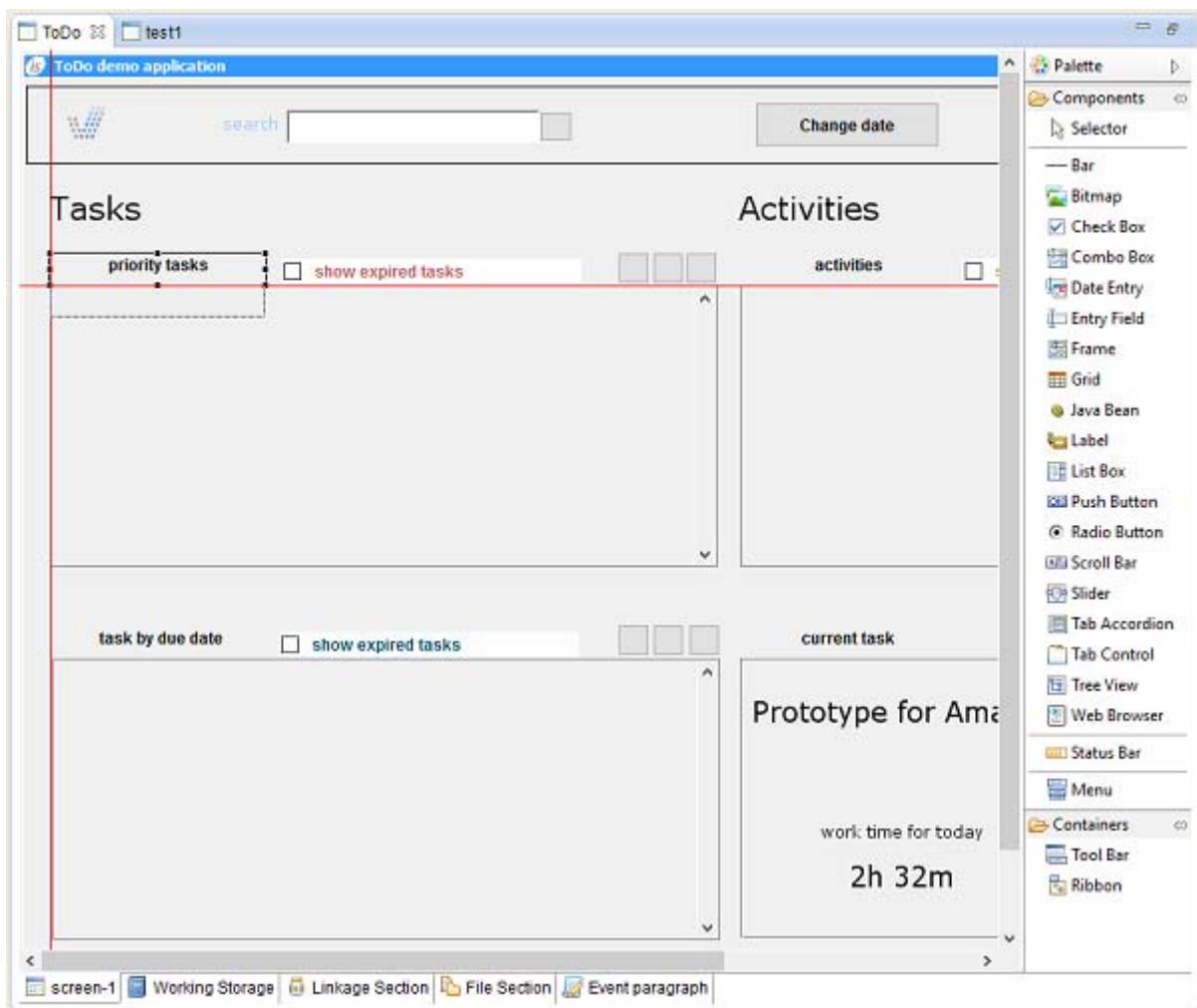
The screenshot shows the isCOBOL IDE interface for service configuration. The top panel, titled "Service Settings", includes fields for "Type: REST", "Prefix: rest", "Response: JSON", and "Entry Point: procedure SONGS". The bottom panel, titled "Data Map", contains two tables: "Linkage Section Fields" and "Service Fields".

Data Item	Value
01 Ink-op-code pic x	
01 Ink-song-data	
05 Ink-sd-id pic 9(5)	
05 Ink-sd-title pic x(30)	
05 Ink-sd-length pic x(5)	
05 Ink-sd-artist pic x(20)	
05 Ink-sd-album pic x(3)	
05 Ink-sd-genre pic x(1)	
05 Ink-sd-label pic x(30)	
05 Ink-sd-year pic 9(4)	
> 05 Ink-sd-authors occur	
< 01 Ink-return-status	

Data Item	Name	Direction	Type
Ink-op-code		input	string
Ink-song-data		input	string
Ink-sd-data		output	string
Ink-sd-id		output	integer
Ink-sd-title		output	string
Ink-sd-length		output	string
Ink-sd-artist		output	string
Ink-sd-album		output	string
Ink-sd-genre		output	string
Ink-sd-label		output	string
Ink-sd-year		output	integer
> Ink-sd-authors		output	string
Ink-return-status		output	string

## isCOBOL IDE Enhancements

The isCOBOL Evolve 2016 R2's IDE includes the new feature "Snap to Guides", which improves productivity by simplifying the alignment of components when designing a screen or report.



## New User Interface Features

The Grid control has been greatly enhanced with new features, and push buttons have been upgraded as well.

### Grid control

Grid content can now be copied to the clipboard and exported in Microsoft Excel XLS and XLSX formats. Those features can be added automatically or controlled by code.

The HEADING-MENU-POPUP property has new values defined in the isgui.def, or automatically generated by the Screen Painter, to allow adding the Copy and Export popup menu items.

Using COBOL code, the new ACTION property value ACTION-EXPORT will trigger the grid data export feature. Exported data file name and format can be customized using the EXPORT-FILE-NAME and EXPORT-FILE-FORMAT properties.

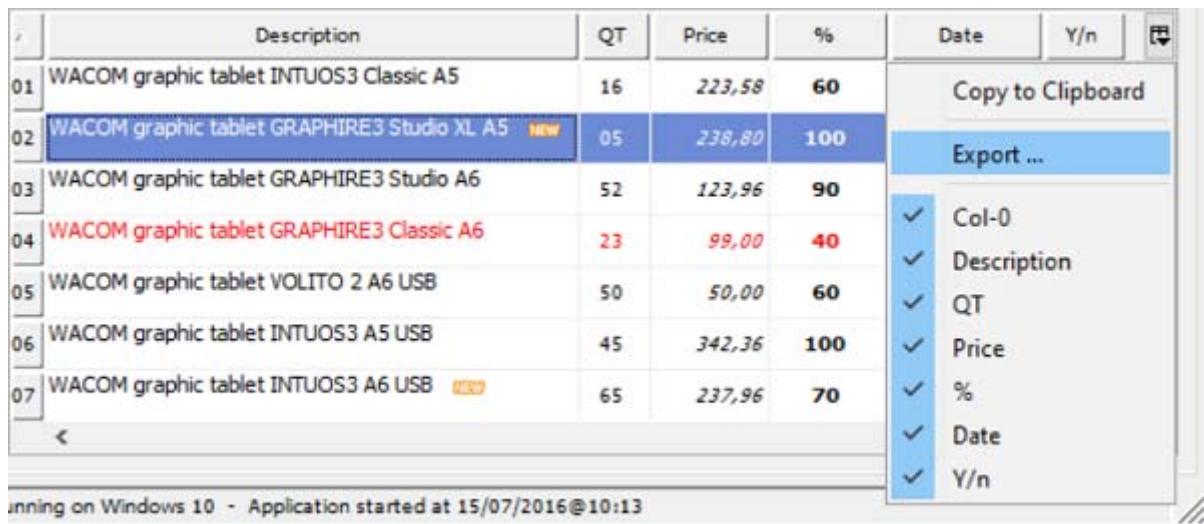
Using the ACTION-COPY value of the ACTION property will copy the grid contents to the clipboard.

The figure below shows how the user can access the new grid export feature. This can be achieved by properly setting the HEADING-MENU-POPUP property of the grid, as shown below, no coding needed!

```
05 my-grid grid
  heading-menu-popup 63
  export-file-name w-path-filename
  export-file-format "xlsx"
```

	Description	QT	Price	%	Date	Y/n	
01	WACOM graphic tablet INTUOS3 Classic A5	16	223,58	60			
02	WACOM graphic tablet GRAPHIRE3 Studio XL A5 <b>NEW</b>	05	238,80	100			
03	WACOM graphic tablet GRAPHIRE3 Studio A6	52	123,96	90			
04	WACOM graphic tablet GRAPHIRE3 Classic A6	23	99,00	40			
05	WACOM graphic tablet VOLITO 2 A6 USB	50	50,00	60			
06	WACOM graphic tablet INTUOS3 A5 USB	45	342,36	100			
07	WACOM graphic tablet INTUOS3 A6 USB <b>NEW</b>	65	237,96	70			
<							

Running on Windows 10 - Application started at 15/07/2016@10:13



Copy to Clipboard

Export ...

✓ Col-0  
✓ Description  
✓ QT  
✓ Price  
✓ %  
✓ Date  
✓ Y/n

The content of exported data into Excel will look like this:

A	B	C	D	E	F	G
1	Description	QT	Price	%	Date	Y/n
2	1 WACOM graphic tablet INTUOS3 Classic A5	16	223,58	60	01/01/2013	0
3	2 WACOM graphic tablet GRAPHIRE3 Studio XL A5	5	238,80	100	01/02/2013	1
4	3 WACOM graphic tablet GRAPHIRE3 Studio A6	52	123,96	90	01/03/2012	0
5	4 WACOM graphic tablet GRAPHIRE3 Classic A6	23	99,00	40	01/04/2013	1
6	5 WACOM graphic tablet VOLITO 2 A6 USB	50	50,00	60	01/05/2013	0
7	6 WACOM graphic tablet INTUOS3 A5 USB	45	342,36	100	01/06/2011	1
8	7 WACOM graphic tablet INTUOS3 A6 USB	65	237,96	70	01/07/2013	1
9	8 WACOM graphic tablet INTUOS3 A4 USB	3	490,68	100	01/08/2013	0
10	9 WACOM Cintiq graphic tablet 17 TFT Vga+DVI	15	2632,56	100	01/09/2013	1
11	10 NGS graphic tablet Draw Master 20x15cm USB	63	59,00	80	01/10/2013	1
12	11 NGS graphic tablet Cadboy 14x10cm USB	14	38,00	10	01/11/2013	0
13	12 WACOM Cintiq graphic tablet 18 TFT Vga+DVI	0	3373,44	100	01/12/2010	1
14	13 WACOM Cintiq graphic tablet 15 TFT Vga+DVI	87	1910,28	50	01/01/2013	0
15	14 WACOM graphic tablet x Notebook PEN PARTNER	14	39,00	100	01/02/2013	1
16	15 COREL CorelDraw Graphic Suite 12 (Upgrade)	63	292,00	100	01/03/2013	1
17	16 COREL CorelDraw Graphic Suite 12 (Full)	55	565,00	90	01/04/2013	0
18	17 COREL CorelDraw Graphic Suite 11 (Full)	23	99,00	40	01/05/2009	1
19	18 PINNACLE Cubasis VST 5.0	45	76,80	60	01/06/2013	1
20	19 ACD SYSTEMS AcdSee 7.0	75	99,90	30	25/07/2013	0
21	20 SONY T2XP/S Centr1.2G 512M 60G DVD±RW 10.6 XP	21	2868,00	50	01/08/2013	1

Multiple selection modes are now supported in the grid control, to allow users to more conveniently select rows or columns.

New properties in the GRID control:

- SELECTION-MODE to specify the selection type
- CELL-SELECTED-COLOR to set the selected cell color, expressed as COBOL value
- CELL-SELECTED-BACKGROUND-COLOR to set the selected cell background color, in RGB format
- CELL-SELECTED-FOREGROUND-COLOR to set the selected cell foreground color, in RGB format
- COLUMN-SELECTED-COLOR to set the selected column color, expressed as COBOL value
- COLUMN-SELECTED-BACKGROUND-COLOR to set the selected column background color, in RGB format
- COLUMN-SELECTED-FOREGROUND-COLOR to set the selected column foreground color, in RGB format
- ROW-SELECTED-COLOR to set the selected row color, expressed as COBOL value
- ROW-SELECTED-BACKGROUND-COLOR to set the selected row background color, in RGB format
- ROW-SELECTED-FOREGROUND-COLOR to set the selected row foreground color, in RGB format
- CELLS-SELECTED to retrieve the selected cells list

- COLUMNS-SELECTED to retrieve the selected columns list
- ROWS-SELECTED to retrieve the selected rows list

With the code shown below multiple row selections can be easily added in the grid:

```
05 my-grid grid
  selection-mode 12
  row-selected-foreground-color rgb x#9CB0E3
  row-selected-background-color rgb x#2D4D9F
  ...
  
```

	Description	QT	Price	%	Date	Y/n
01	WACOM graphic tablet INTUOS3 Classic A5	16	223,58	60	01/01/2013	<input type="checkbox"/>
02	WACOM graphic tablet GRAPHIRE3 Studio XL A5 NEW	05	238,80	100	01/02/2013	<input checked="" type="checkbox"/>
03	WACOM graphic tablet GRAPHIRE3 Studio A6	52	123,96	90	01/03/2012	<input type="checkbox"/>
04	WACOM graphic tablet GRAPHIRE3 Classic A6	23	99,00	40	01/04/2013	<input checked="" type="checkbox"/>
05	WACOM graphic tablet VOLITO 2 A6 USB	50	50,00	60	01/05/2013	<input type="checkbox"/>
06	WACOM graphic tablet INTUOS3 A5 USB	45	342,36	100	01/06/2011	<input checked="" type="checkbox"/>
07	WACOM graphic tablet INTUOS3 A6 USB NEW	65	237,96	70	01/07/2013	<input checked="" type="checkbox"/>
08	WACOM graphic tablet INTUOS3 A4 USB	03	490,68	100	01/08/2013	<input type="checkbox"/>
09	WACOM Cintiq graphic tablet 17 TFT Vga+DVI	15	2632,56	100	01/09/2013	<input checked="" type="checkbox"/>
--	I-MCS graphic tablet Draw Master 20x15cm LCD	--	--	--	--	<input type="checkbox"/>

Running on Windows 10 - Application started at 17/06/2016@14:31

To provide better looking and easier to read grids, when multiple header rows are used, heading cells can now span horizontally or vertically

- CELL-ROWS-SPAN spans a header cell vertically on multiple rows
- CELL-COLUMNS-SPAN spans a header cell horizontally on multiple columns

In the figure below, the SPAN CELLS feature is used to vertically span the first 4 columns and to horizontally span the Album Info cell, using the following code:

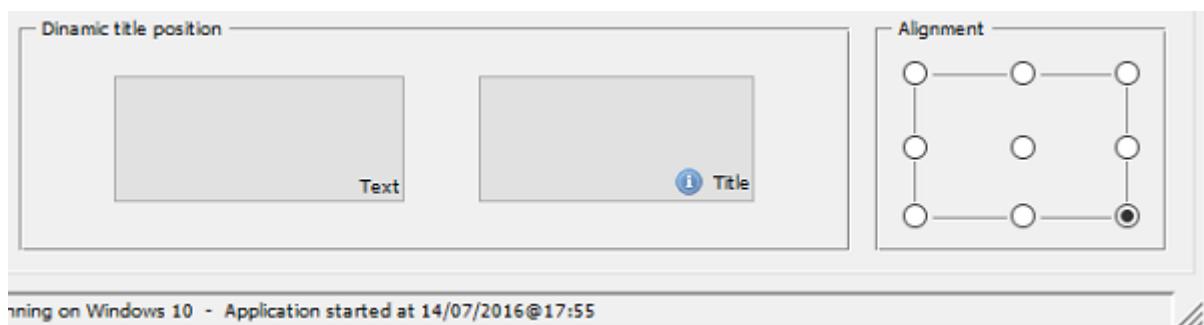
```
05 my-grid grid
  column-headings
  num-col-headings 2
  ...

  modify my-grid(1, 1) cell-rows-span 2
  modify my-grid(1, 2) cell-rows-span 2
  modify my-grid(1, 3) cell-rows-span 2
  modify my-grid(1, 4) cell-rows-span 2
  modify my-grid(1, 5) cell-columns-span 3
  
```

	Title	Length	Artist	Album Info		
				Name	Genre	Label
01	Let It Be	4:03	Beatles	Let It Be	Pop	Apple Records
02	Yellow Submarine	2:40	Beatles	Revolver	Pop	Apple Records
03	Help!	2:21	Beatles	Help!	Pop	Parlophone
04	Yesterday	2:07	Beatles	Help!	Pop	Parlophone
05	Angie	4:30	The Rolling Stones	Goats Head Soup	Rock	R.S.Records
06	Start Me Up	3:32	The Rolling Stones	Tattoo You	Rock	Rolling Stones records

## Push buttons

Alignment styles LEFT, RIGHT, TOP, BOTTOM, CENTER are now supported in push-buttons and can be set dynamically.



## Framework Improvements

The isCOBOL framework has yet again been improved to aid developers gain productivity while debugging and increase performance of running applications.

### C\$WRITELOG

Multiple parameters are now supported in C\$WRITELOG, to output several items at once, as shown below.

```
CALL "C$WRITELOG" using "value of var1:" var1 ", var2=" var2
```

### W\$FLUSH

The W\$FLUSH library routine has two new op-codes:

- WFLUSH-DISABLE-UI to disable the user interface updates
- WFLUSH-ENABLE-UI to restore the user interface updates

The new opcodes can improve performance in scenarios where a long computation contains several unnecessary DISPLAY or MODIFY statements. Disabling the UI drawing will speed up performance and, at the end of the computation, UI drawing can be re-enabled to update the user interface. It can be used as shown below.

```
CALL "W$FLUSH" using WFLUSH-DISABLE-UI
PERFORM DO-LONG-OPERATION-WITH-UNNECESSARY-UI-UPDATES
CALL "W$FLUSH" using WFLUSH-ENABLE-UI
```

## External logging

External logging can now be configured using the new configuration property:

```
iscobol.logclass=com.iscobol.logger.Slf4jLogger
```

This allows the use of external logging libraries, such as Log4J, and take advantage of their advanced features, like rolling and zipping.

## isUPDATER

isUPDATER can now automatically check for software updates by adding the -update option in the command line used to run an isCOBOL application, such as

```
iscrun -update MYPROG
```

isUPDATER loads from CLASSPATH the isupdater.properties configuration file, containing information on the update server, such as

```
swupdater.site=http://192.168.0.123:10996
```

using this configuration, isUPDATER is leveraging the -hs option of iscserver, which starts an HTTP server on the default 10996 port, where updates might have been stored.

## isCOBOL Compiler Enhancements

isCOBOL Evolve 2016 R2 includes several changes to the isCOBOL Compiler that improve productivity and simplify migration from other COBOLs.

### \$SET to set compiler properties on each program

\$SET directive can now be used to set compiler properties for each program, to allow customization of compiler properties inside the source code, without the need for additional configuration files, such as:

```
$set "easylinkage" "1"
$set "easylinkage.package" "com.veryant"
PROGRAM-ID. GETCUSTID.
```

### Enhanced compatibility with other COBOLs

New library routines have been implemented: CBL\_EQ and CBL\_IMP for logical operator, CBL\_SPLIT\_FILENAME to split a filename. New intrinsic functions have been added: E, EXP, EXP10, FRACTION-PART, PI, SIGN to simplify migration from Micro Focus COBOL.

Vision version 6 indexed files are now supported in the com.iscobol.io.ScanVision file handler – used in ISMIGRATE - to allow data migration from ACUCOBOL-GT Extend 10.

A new configuration property, iscobel.gui.screen\_col\_zero=1, has been added to emulate the RM/COBOL behavior of the DISPLAY statement with the COLUMN 0 phrase.

ESQL TRUNCATE statement is now supported to simplify migration from Pro\*COBOL to isCOBOL with JDBC database access.

## IsCOBOL Server Improvements

isCOBOL Thin Client can now be updated without needing any client configuration changes. By properly configuring isCOBOL Server, isUPDATER automatically updates client components.

Client updates can now be configured on isCOBOL Server, using the new iscobel.as.clientupdate.site property, by declaring the HTTP server location where updates are located, such as

```
iscobel.as.clientupdate.site=http://192.168.0.123:10996
```

The above configuration takes advantage of the -hs option of isCOBOL Server which starts an http server on port 10996. Client components are then guaranteed to match the server version.

If minor server updates are not wanted on the clients, these can be skipped by setting the client runtime version to the desired value using the new server configuration property iscobel.as.clientupdate.version, i.e.

```
iscobel.as.clientupdate.version=875.2
```

Single clients can be configured to skip updates with the new client option *-noupdate*

```
iscclient -hostname ipserver -port 10999 -noupdate MYPROG
```

## isCOBOL EIS improvements

### Servlet prefix

You can now customize the prefix used by the servlet to map the web service operation to the program generated by Service Bridge, by setting the iscobel.http.servlet.prefix configuration property.

For example, if the Service Bridge was used to generate a Rest web service, the prefix must be configured as

```
iscobel.http.servlet.prefix=rest
```

while if a SOAP web service was generated, then the prefix should be

```
iscobel.http.servlet.prefix=soap
```

If Service Bridge was configured with custom prefixes when the program was generated, then you need to specify the same prefix

```
iscobel.http.servlet.prefix=custom
```

## Utility improvements

### isUPDATER enhancements

isUPDATER has been enhanced to supports folder names for updates in addition to zip files. Native, OS specific updates, can be downloaded as needed by appropriately setting the isUPDATER configuration file. This will save time and bandwidth, by allowing the download of only necessary components.

An example of multiple OS configuration settings:

```
swupdater.version.iscobol=875.2
swupdater.lib.iscobol=lib
swupdater.version.iscobolNative=875.2
swupdater.lib.linux.32.iscobolNative=native/linux32_libs
swupdater.lib.linux.64.iscobolNative=native/linux64_libs
swupdater.lib.win.32.iscobolNative=win32_libs
swupdater.lib.win.64.iscobolNative=win64_libs
```

# isCOBOL 2016 Release 1 Overview

## Introduction

Veryant is pleased to announce to selected users, the latest release of isCOBOL™ Evolve, isCOBOL Evolve 2016 R1.

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

isCOBOL 2016 R1 includes a new product named isUPDATER, which allows you to keep all COBOL application, modules and isCOBOL framework installed up to date.

isCOBOL 2016R1, also includes the ability to generates Excel file format from isCOBOL IDE generated report as well a native Android SDK integration to better develop mobile application. An even better isCOBOL Debugger with several usability improvements, better runtime performance in several areas, and new UI features to continue to satisfy developer requests.

Details on these enhancements and updates are included below.

## isUPDATER, software updates for all COBOL applications

Updated software versions and program updates are vital steps in staying secure online, as security patches are often included in updates, as well as solutions for other vulnerabilities and bugs. The goal of this new product, is to assist you to keep all COBOL applications, native modules and isCOBOL libraries up to date.

The isUPDATER software:

- Provides an easy, one-click activation of applications
- Guarantees that you are always running the latest version of the application
- Eliminates complicated installation or upgrade procedures

Based on HTTP communication protocol it can be used on conjunction with any HTTP server live IIS or Apache. In order to simplify the use of this new feature, isCOBOL Server provides very basic HTTP services to make the adoption of this new feature out-of-the box.

With the server configuration file, it is possible to define which packages that must be keep up to date. It is also possible to define a client-side POSTUPDATE procedure for each package to be updated. This allows the developer to specify specific tasks to be executed on the client where packages are updated, after updating procedures.

On server side we need to have a property file named swupdater.properties where define packages and versions:

```
swupdater.version.iscobel=100  
swupdater.zipfile.iscobel=isCOBOL2016.zip  
swupdater.version.application=100  
swupdater.zipfile.application=customer_application.zip
```

to be located with all packages for updating process on a directory of a HTTP server like Apache Http, Microsoft IIS or isCOBOL Server with HTTP service tuned on.

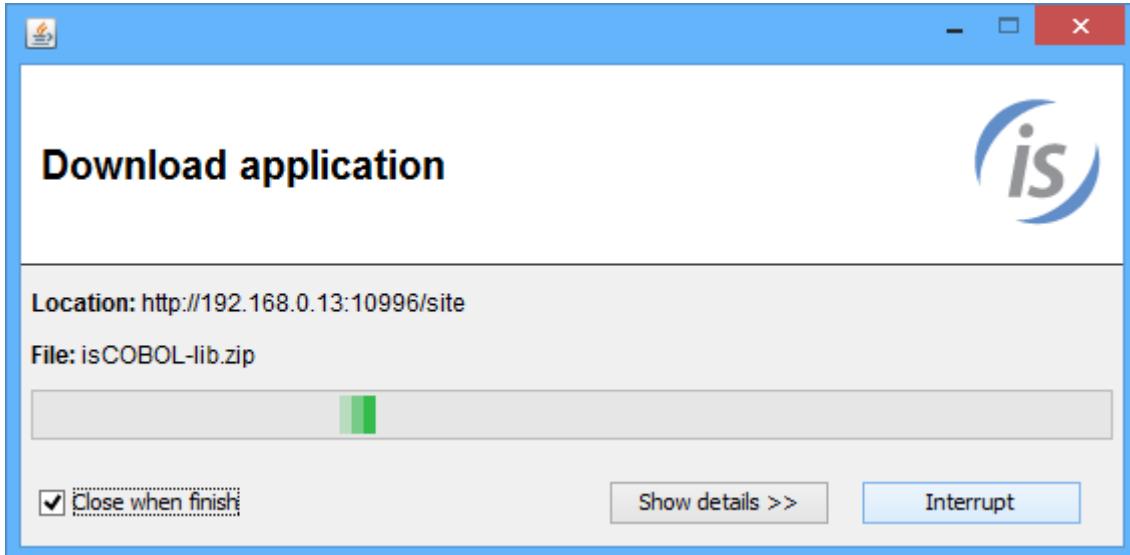
On the client site we need to have a property file, named for example isupdater.properties, where defines mainly the address of HTTP server where locate updates and packages to be updated:

```
swupdater.site=http://192.168.0.13:10996/site  
swupdater.version.iscobel=90  
swupdater.directory.iscobel=c:/isCOBOL2016R1  
swupdater.version.application=90  
swupdater.directory.application=C:/Application  
swupdater.mainclass=MAIN_APPLICATION
```

With the above basic configuration we can run the isUPDATER command on client site to start the updating process from a command prompt typing:

```
iscupdater -c isupdater.properties
```

This command will launch the updating process as ruled on server and client configuration file. As depicted in the picture below, the command will show a dialog where will be possible to see all actions of updating procedure.



After updating the packages, as defined with swupdater.mainclass property, will be executed the main program of COBOL application named MAIN\_APPLICATION.

Another use of the updater facility could be to run isCOBOL thin client afterwards to accommodate the updating of jar files on the client side. This configuration will remove the task to update the jar files on the client side when the version of the isCOBOL Server doesn't match the version of isCOBOL thin client.

The following property file allows running the thin client after client's updating:

```
swupdater.site=http://192.168.0.13:10996/site  
swupdater.version.iscobel=90  
swupdater.directory.iscobel=c:/isCOBOL2016R1  
swupdater.version.application=90  
swupdater.directory.application=C:/Application  
swupdater.mainclass=com.iscobel.gui.client.Client MAIN_APPLICATION
```

## isCOBOL IDE Enhancements

In isCOBOL Evolve 2016 R1, the isCOBOL IDE includes changes that improve the usability and productivity with additional new features like Excel file export from the Report Designer generated program.

### Report Designer

isCOBOL IDE 2016 R1 enhanced the Report Designer adding the ability for report programs to export into Excel file formats. As depicted in the picture below, during print preview, the user can decide to export the output generated from print program to XLS or XLSX file formats as well to generate a PDF or to print directly to the physical printer.

Print Preview

Export

Page 1 of 2



2015/12/22 04:36

Code	Name	Description	Price
<b>Products for Brand:</b>			
AcdSee	AcdSee 7.0	ACD SYSTEMS AcdSee 7.0	1,050.50
			<b>Brand Sub-total</b>
			<b>1,050.50</b>
<b>Products for Brand:</b>			
COREL1	CorelDraw 12 (Upg)	COREL CorelDraw Graphic Suite 12 (Upgrade)	23,410.50
COREL2	CorelDraw 12 (Full)	COREL CorelDraw Graphic Suite 12 (Full)	43,210.50
COREL3	CorelDraw 11 (Full)	COREL CorelDraw Graphic Suite 11 (Full)	3,104.50
			<b>Brand Sub-total</b>
			<b>69,725.50</b>
<b>Products for Brand:</b>			
CUB	Cubasis VST 5.0	PINNACLE Cubasis VST 5.0	1,043.50
			<b>Brand Sub-total</b>
			<b>1,043.50</b>

The next picture shows an Excel sheet created automatically from the new Export menu of print preview. As you can note, the export feature converts all graphical definition like fonts, formats and images into Excel equivalents features:

A1	B	C	D	F	G	H	I	J	K
1									
2									
5									
6									
7	Code	Name		Description					Price
8									
9	Products for Brand:			AcdSee					
11	AcdSee	AcdSee 7.0		ACD SYSTEMS AcdSee 7.0					1,050.50
12									
13							Brand Sub-total		1,050.50
14									
15	Products for Brand:			COREL					
17	COREL1	CorelDraw 12 (Upg)		COREL CorelDraw Graphic Suite 12 (Upgrade)					23,410.50
19	COREL2	CorelDraw 12 (Full)		COREL CorelDraw Graphic Suite 12 (Full)					43,210.50
21	COREL3	CorelDraw 11 (Full)		COREL CorelDraw Graphic Suite 11 (Full)					3,104.50
22							Brand Sub-total		69,725.50
23									
24									
25	Products for Brand:			Cubasis					
27	CUB	Cubasis VST 5.0		PINNACLE Cubasis VST 5.0					1,043.50
28							Brand Sub-total		1,043.50
29									

To better configure the output layout generations, the Export feature defaults and behaviors can be adjusting using the following properties:

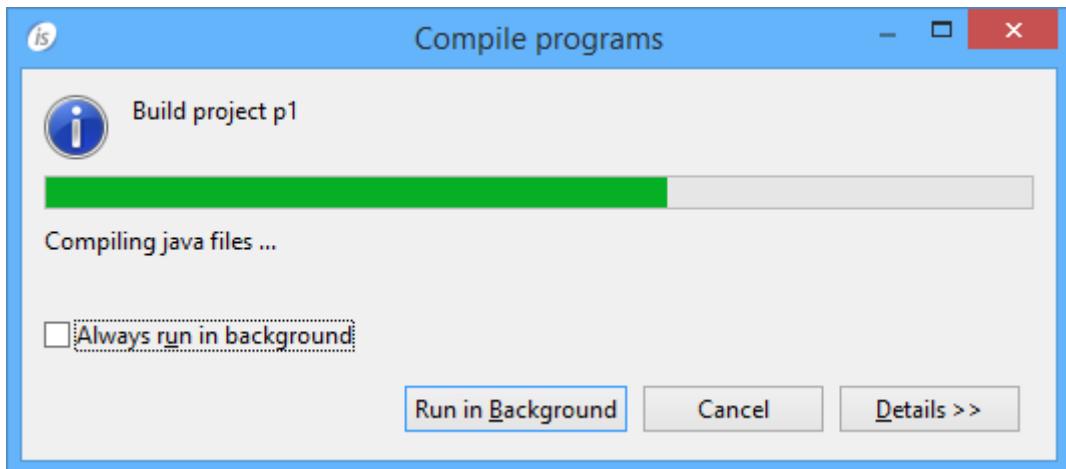
- iscobel.export.excel.cell\_ignore\_background=(true|false)
- iscobel.export.excel.cell\_ignore\_borders=(true|false)
- iscobel.export.excel.cell\_locked=(true|false)
- iscobel.export.excel.cell\_numeric\_format=(valid excel numeric format)
- iscobel.export.excel.cell\_wrap\_text=(true|false)
- iscobel.export.excel.collapse\_row\_span=(true|false)
- iscobel.export.excel.detect\_cell\_type=(true|false)
- iscobel.export.excel.force\_page\_breaks=(true|false)
- iscobel.export.excel.freeze\_page\_header=(true|false)
- iscobel.export.excel.ignore\_images=(true|false)

- iscobel.export.excel.remove\_columns\_space=(true|false)
- iscobel.export.excel.remove\_rows\_space=(true|false)
- iscobel.export.excel.whitepage\_background=(true|false)

### Ability to run the compilation in background mode

Added the ability to run the compilation on one or more programs in background mode.

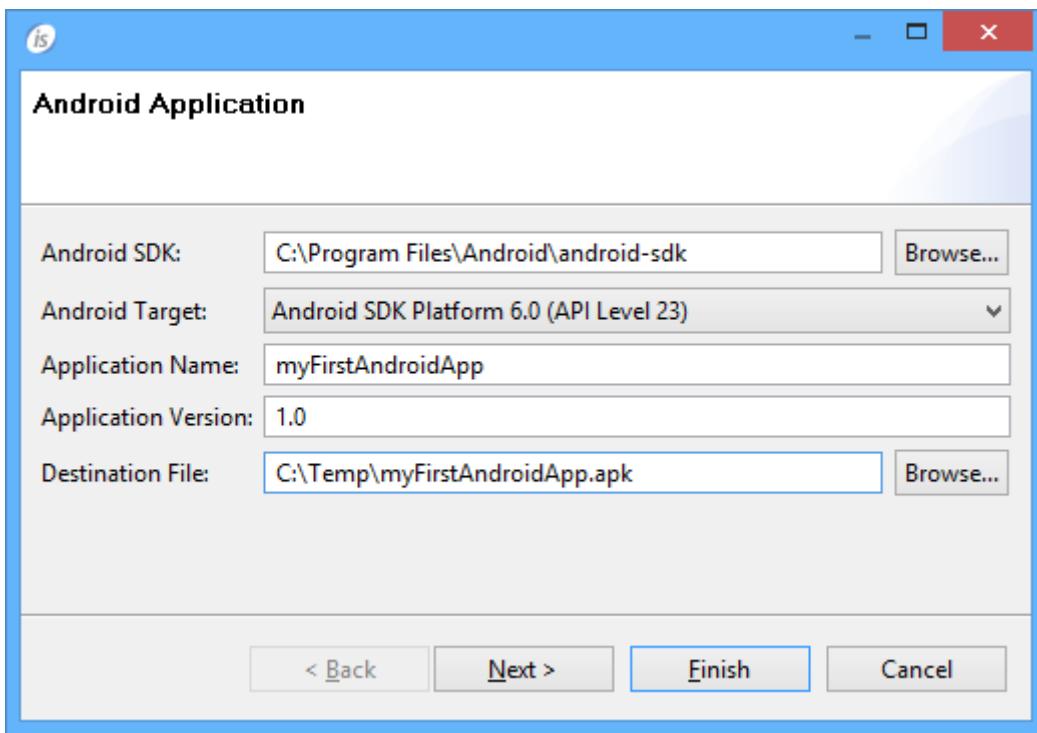
This allows during long compilations to continue working on other items in the project without wasting the time spent during compilation (like shown in the picture below).



### ANDROID SDK integration

In order to simplify the mobile application generation, isCOBOL 2016R1 natively supports the generation of Android packages. As depicted in the picture below, a new Wizard named "Export to Android Application" is available in isCOBOL HTML project of isCOBOL IDE.

This new step by step wizard allows to create the APK without any external Eclipse plugin installed, just fill required fields to specify the SDK to be used, the application names, the application version etc.



## Debugger Improvements

isCOBOL Debugger continues to be a core feature of isCOBOL Evolve so we like the idea to continue to improve it. For this reason in isCOBOL 2016 R1, the isCOBOL Debugger provides new features to continue to improve usability and easy to be used.

### New Current Variables View

isCOBOL Debugger now automatically shows the Current Variables to easily follow the program execution step by step. As depicted in the picture below, in the right part of Debugger window, an area is reserved to show the Current Variables (list of variables used in the current and previous statement) and the Watched Variables, so it is more simple to see multiple variables during the stepping process.

The screenshot shows the isCOBOL Graphic Debugger interface. The main window displays a COBOL program named ISCONTROLSET.cbl. The code includes several ACCEPT statements for system information and a PERFORM loop. The Variables panel on the right shows the current values of variables like WTIME and TODAY, along with a list of watched variables under SYSTEM- INFORMATION and TERMINAL-ABILITIES. The bottom status bar indicates the current paragraph and location.

```

2710      screen line win-line.
2711
2712
2713  RETRIEVE-SYSTEM- INFORMATION.
2714    accept today from century-date
2715    accept wtime from time
2716    accept system-information from system-info
2717    accept terminal-abilities from terminal-info.
2718
2719    perform RETRIEVE-SERVER- INFORMATION
2720
2721  if is-remote
2722    perform RETRIEVE-CLIENT- INFORMATION
2723  end-if.
2724
2725  RETRIEVE-SERVER- INFORMATION.
2726
2727  call "J$NETADDRESS" using s-hostname s-hostip
2728
2729  call "C$GETENV" using "java.vm.name"
2730          s-java-tm-name
2731
2732  call "C$GETENV" using "java.version"

```

Name	Value	Hex
WTIME	14344351	3134333434333551
TODAY	20151202	3230213591223032

Name	Value	Hex
SYSTEM- INFORMATION		
TERMINAL- ABILITIES		
--TERMINAL- NAME	xterm	78740572002
--FILLER	Y	59
--FILLER	N	4E
--FILLER	Y	59
--FILLER	Y	59
--FILLER	N	4E
--FILLER	Y	59
--FILLER	Y	59

## Other minor enhancements

The picture below, shows the new management introduced in debugger interface to better manage long values in tool-tip and monitor.

This screenshot shows the same isCOBOL Graphic Debugger interface, but with a focus on handling long character strings. In the Variables panel, the variable VAR1 is shown with its full value (a long string of 'A's) displayed in a tooltip, demonstrating improved management of such values. The bottom status bar also reflects this change.

```

1  program-id. PROG.
2
3  VAR1 = AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
4
5
6  pro
7  mai
8
9
10 move " B" to var1(1000:2)
11 continue
12 goback.
13
line=10 file=PROG.cbl
move " B" to var1(1000:2)
line=11 file=PROG.cbl
continue

```

Name	Value
VAR1 [PROG]	AAAAAA[REDACTED]AAAAAA

A New button (green plus) was added in Monitor and Breakpoint Debugger views to simplify the creation of new Monitor or Breakpoint. Additionally, a new configuration property named `iscobol.debug.propfile` was added to set a different property file to store Debugger settings to simplify for a team of to use the same settings.

## New User Interface Features

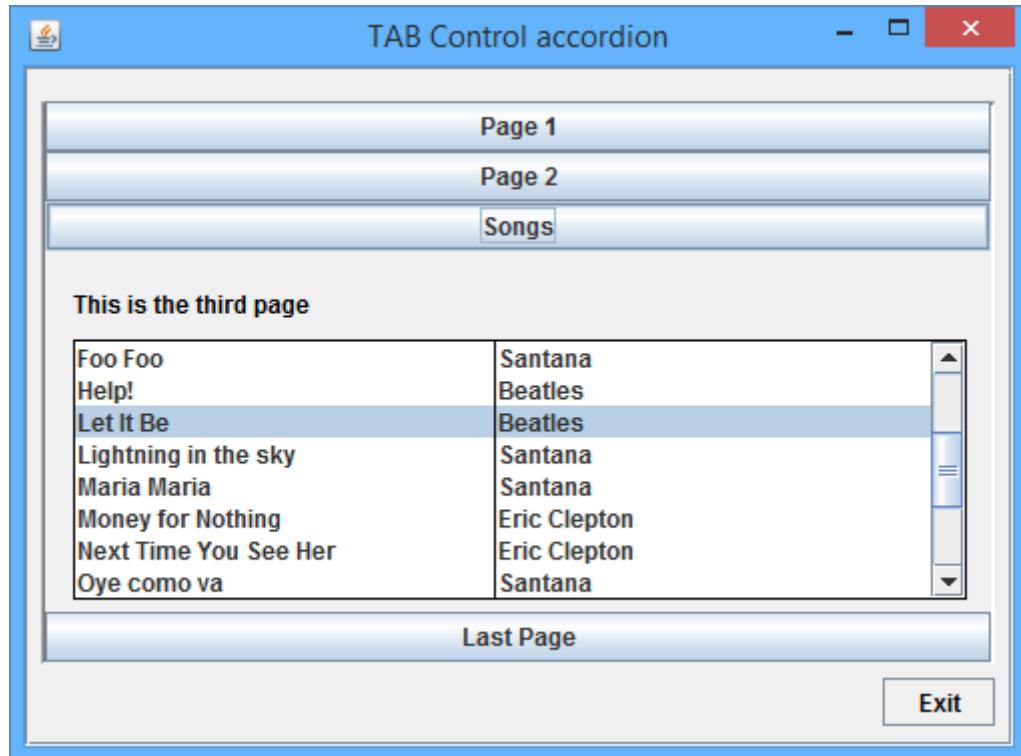
To stay up to date with new requests to develop modern and easy to use applications, Veryant engineers continue to improve UI with new controls, style and properties. An Example? The ACCORDION new style in tab-control.

### New layout ACCORDION in TAB-CONTROL

isCOBOL 2016 R1 introduces a new layout in TAB-CONTROL. This layout can be activated with the new style named ACCORDION. As per the picture below, it is clear that ACCORDION is useful to groups some controls that will be visible only when selecting item (page) will be clicked.

Code example:

```
03 Tb1-accordion  
  tab-control  
    line 2 col 2 lines 17 cells size 68 cells  
    ACCORDION.
```



## New control properties

New property in ENTRY-FIELD control:

- PROPOSAL-MIN-TEXT to show the proposals after a number of characters typed in the field. This works in conjunction with existing PROPOSAL properties.

New property in ENTRY-FIELD and COMBO-BOX controls:

- PLACEHOLDER to show a hint inside the control when the field is empty. This assist the User to understand which type of information is required in the editing field.

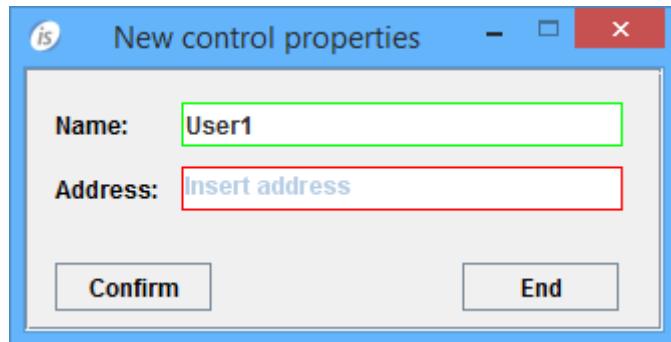
New property available in all controls with borders:

- BORDER-COLOR sets the border color when BOXED style is set (in entry-field it is set by default). It supports all color values and also RGB colors.

The following example shows how to use the new properties PROPOSAL-MIN-TEXT, PLACEHOLDER and BORDER-COLOR:

```
03 efname
  entry-field line 2 col 12 size 30
  border-color 11
  placeholder "Insert name"
  proposal-min-text 2.
03 efaddress
  entry-field line 4 col 12 size 30
  border-color 13
  placeholder "Insert address".
```

The image below shows the final result of BORDER-COLOR on the fields and the PLACEHOLDER see just on the second field because it is still empty.



## Framework Improvements

isCOBOL framework is the heart of isCOBOL Evolve. Once again Veryant's engineers gain performance on many common statements to perform faster the whole COBOL application. In addition of performance improvements, isCOBOL 2016R1 provides JISAM encryption, reduction of the memory footprint on numeric fields and many other improvements.

## Performance and memory improvements

Many COBOL statements have been optimized in order to increase execution of batch COBOL programs. To take advantage of most performance improvements, the COBOL source code should be recompiled with isCOBOL compiler 2016R1.

This is the list of statements that were improved in 2016R1:

- ADD,
- EVALUATE,
- EXIT SECTION,
- IF,
- MOVE,
- PERFORM,
- SET,
- SUBTRACT.

2016R1 also optimizes COBOL programs that use C memory model (-cp) on several areas. The memory consumed for numeric COBOL variables allocation has been reduced.

The image below shows a comparison between isCOBOL 2016R1 and isCOBOL 2015R1. The performance test was executed on Windows 10 64 bit (CPU: Intel Core i-5 Processor 4440+, 3.10 Ghz / RAM: 8GB / isCOBOL compiler option -dz / java used: Oracle JDK1.8.0\_65 with option –server, time in seconds).

Statements	isCOBOL 2015 R1	isCOBOL 2016 R1
if varx = "A"	0,38	0,37
if varnum = 1	0,20	0,16
if var(1:1) = "A"	1,38	0,53
if var(woffset:1) = "A"	1,77	0,67
if var(1:wsize) = "A"	1,79	0,69
if var(woffset:wsize) = "A"	2,09	1,01
move "A" to varx	0,57	0,40
move 1 to varnum	0,52	0,26
move x1 to var(1:1)	3,15	0,54
move x1 to var(woffset:1)	3,38	2,09
move x1 to var(1:wsize)	3,48	2,15
move x1 to var(woffset:wsize)	3,67	2,28
evaluate numvar 100 when	3,47	1,18
add 1 to varnum18	2,97	1,33
subtract 1 from varnum18	2,99	1,34
perform varying varnum from 1 by 1	3,11	1,46
perform with exit section	0,15	0,13
initialize table with nested occurs	3,98	3,65
set 88level to true	1,87	0,24
set varnum to size of varx	0,11	0,09
call prog using C\$PARAMSIZE	2,54	2,11
Total:	43,57	22,68

## Encryption on JISAM files

isCOBOL compiler 2016 R1 supports the WITH ENCRYPTION clause in the SELECT of indexed files. This clause was managed as a comment in previous releases to turn on a compression based on public and private keys. A new property named:

- iscobel.file.encryption.key

Allows to specify the key to be passed to file handler. This property can be set dynamically inside COBOL programs using SET ENVIRONMENT statement to maximize security.

Code example:

```
select pwd-file assign to "pwdfile"
      organization is indexed
      access mode is dynamic
      record key is pwd-key
      WITH ENCRYPTION
```

When it runs, all data stored in the "pwdfile" (both .dat and .idx) will be encrypted based on the encryption value.

In case the configuration is missing, a new file status "9X" will be returned when opening an encrypted file that means "Encryption's key missing".

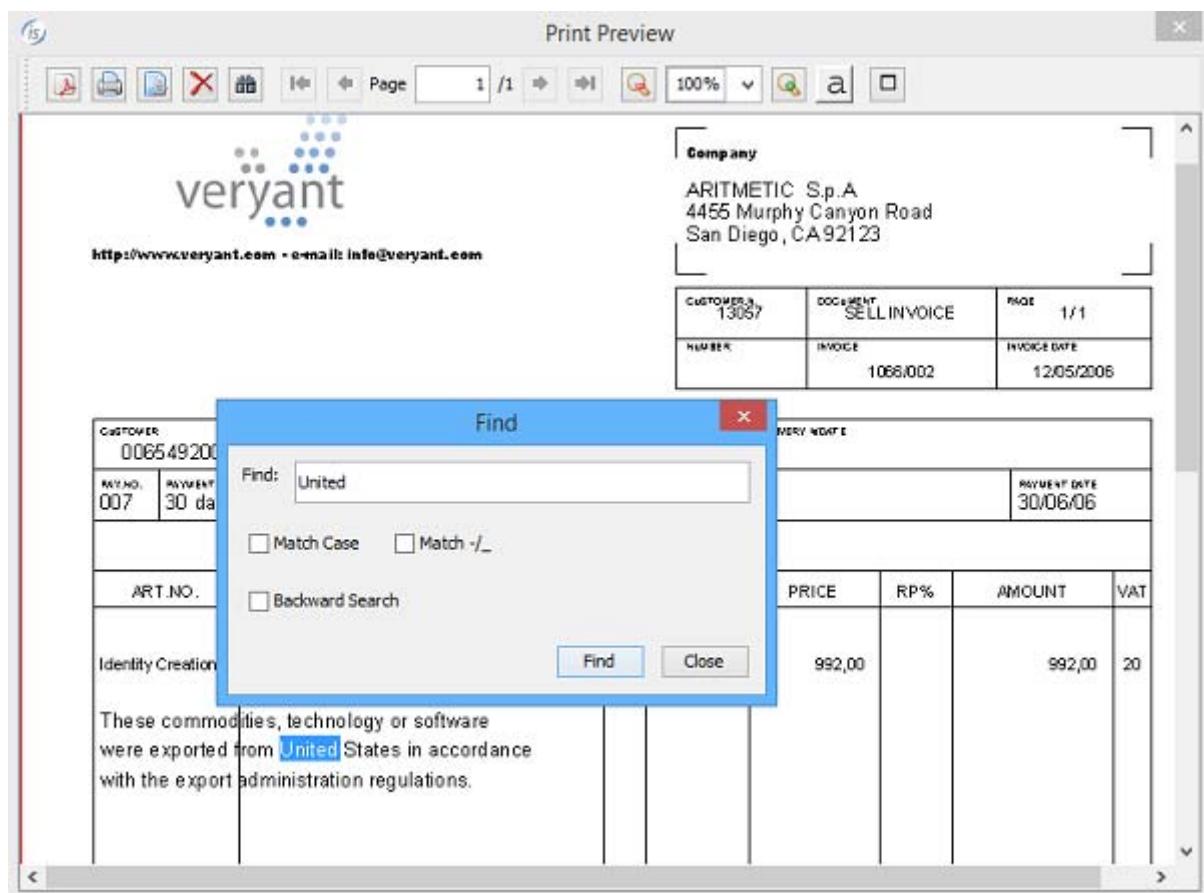
JUTIL command line utility was updated to work correctly with encrypted JISAM files. A new option -e=encryption\_key needs to be passed to have JUTIL working correctly on encrypted files.

JUTIL command line examples:

```
jutil -e=MyKey123 -unload pwdfile outputbinpwd  
jutil -e=MyKey123 -rebuild pwdfile
```

### Find features on print preview

As depicted the picture below, the "find" feature is now supported in the isCOBOL Print Preview. User of print preview can now easily search text in forward or backward way with the option to match -/\_.



### Ability to know who is locking a record under c-tree

A new library routine named "C\$LOCKPID" has been implemented to retrieve from COBOL program who is locking a record with c-tree file handler.

It returns the c-tree task number to be used in the Faircom utilities to monitor the connected clients.

Code example:

```
77 wctid pic 9(7).
...
read myfile next with lock
if myfile-fs = "99"
  call "C$LOCKPID" giving wctid
end-if
```

## Other enhancements

Here some minor enhancements provided in isCOBOL 2016R1 framework:

- Ability to have all ACCEPT statements to time out just as if there was a BEFORE TIME phrase present in the ACCEPT statement. A new configuration property iscobel.accept\_timeout to specify the number of seconds to timeout. Configuration example:

```
iscobel.accept_timeout=5
```

- Ability to set attributes for PDF generation through configuration properties. The syntax is iscobel.print.attribute.attribute-name=attribute-value.

The attribute-name and attribute-value are the same used from WINPRINT-SET-ATTRIBUTE op-code in WIN\$PRINTER library routine.

Configuration examples to define jpeg compression to 95% and PDF author to "MyName":

```
iscobel.print.attribute.jpeg=95
iscobel.print.attribute.author=MyName
```

- Improved PDF compression especially when the same images are used many times in the print job.

## isCOBOL Compiler Enhancements

isCOBOL Evolve 2016 R1 release, includes many changes on the isCOBOL Compiler that improve its power and flexibility and simplify some areas of migration to isCOBOL from other COBOLs and Java interoperating.

### Object Oriented syntax check improvements

A new compiler Warning was added for method invocation used without cast when the signature is ambiguous. This new Warning helps to identify lines that could be better written using the cast to be sure hitch on the method that will be invoked when running.

Compiling the code in the following example:

```
repository.
  class math as "java.lang.Math"
working-storage section.
 77 w-result pic 9(18).
procedure division.
  set w-result = math:>max(11, 22)
```

will return this compiler Warning on the line where “max” method is invoked:

```
--W: #222 The method signature is ambiguous, use a (better) cast: max;
```

This occurs because the java class has 2 methods with the same name and same number of parameters but different type:

```
static int max(int a, int b)
static long max(long a, long b)
```

To avoid the Warning the programmer can easily set which cast needs to be applied on the method invocation. In the previous example:

```
set w-result = math:>max(11 as long, 22 as long)
```

or

```
set w-result = math:>max(11 as int, 22 as int)
```

## EasyLinkage feature to simplify calling isCOBOL from Java

Some new compiler properties are added to generate a java bridge program that allows to easily run an isCOBOL program from Java. The EasyLinkage feature simplify the previous usage of invoke isCOBOL programs from Java because now there is no need for Java developer to know the cobol variable type and structure of parameters in Linkage Section.

Considering to have the linkage section inside a cobol program named PROG1 defined in this way:

```
linkage section.
 77 P1 pic 9(5) comp.
 77 P2 pic x(20).
 procedure division using p1 p2.
 ...
```

The compiler automatically generate a java class named linkPROG1, so this “bridge class” will be used from a Java source in this way:

```
public class test {
    public static void main (String[] args) throws Exception {
        //create an instance of linkPROG1
        linkPROG1 prog1 = new linkPROG1();
        //set the p1 parameter to 123 and p2 parameter to "ABC"
        prog1.p1.set(123);
        prog1.p2.set("ABC");
        //do the call
        prog1.run();
        //show the returned parameters
        System.out.print("p1=" + prog1.p1.toInt());
        System.out.print("p2=" + prog1.p2.toString());
    }
}
```

To better configure the output generations of EasyLinkage feature, isCOBOL compiler support the following properties:

- iscobel.compiler.easylinkage=true to generate the bridge program
- iscobel.compiler.easylinkage.prefix=prefix to use a different prefix (default "link")
- iscobel.compiler.easylinkage.package=packagename to generate with the package
- iscobel.compiler.easylinkage.cut=<val1> <...> to cut val1 prefix in the linkage variable names
- iscobel.compiler.easylinkage.decoration=false to avoid the use of decorated variable names

## Enhanced compatibility with other COBOLs

The isCOBOL 2016 R1 compiler introduces additional new options:

- -d1 to treats Binary data whose length is <= 2 to be stored in 1 byte. It could be useful during Micro Focus or other COBOL migrations when binary files are stored with this rule.
- -dcr to use Realia COBOL sign encoding. It could be useful when the sign encoding used in existing files must be maintained.

The library routine named C\$XML was enhanced to support two new op-codes to simplify migration from ACUCOBOL-GT. This library should be used just for compatibility with ACUCOBOL-GT COBOL source code. isCOBOL users can take advantage of powerful and native way to support XML streams.

Two new library routines named CBL\_WRITE\_SCR\_CHARS and CBL\_READ\_SCR\_CHARS were added to simplify migration from Micro Focus COBOL.

DISPLAY POP-UP statement with CONTROL clause syntax has been enhanced to simplify migration from RM/ COBOL.

Code example:

```
DISPLAY POP-UP WINDOW-CONTROL-BLOCK
      LINE W-LINE POSITION W-POS
      CONTROL W-CONTROL.
ACCEPT W-STAT FROM EXCEPTION STATUS.
CLOSE POP-UP WINDOW-CONTROL-BLOCK CONTROL "WINDOW-REMOVE".
```

Starting from isCOBOL 2016, BTRIEVE and Pervasive ISAM file are directly supported from a new interface class named com.iscobol.io.DynamicBtrieve. This native interface requires installation of BTRIEVE or Pervasive SQL native client library to work correctly.

ISMIGRATE data migration utility was enhanced to support direct BTRIEVE/Pervasive SQL data migration.

## IsCOBOL Server Improvements

Starting from isCOBOL 2016R1, isCOBOL Server was improved to provide more functionality and better performance.

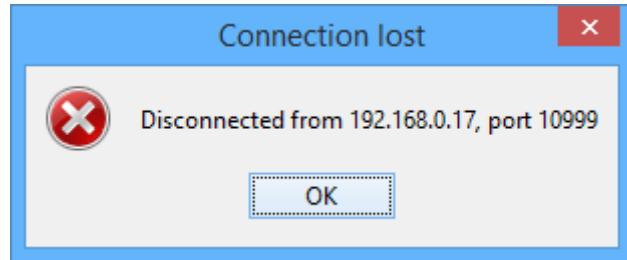
### Lost connection message

A message box 'Lost connection' is now automatically reported on Thin Client architecture if an abnormal session termination is detected. This assist end-users to identify a problem related to the network stability.

In case you need to avoid this message, it's possible to pass the new Client option -nodosconnecterr. Example:

```
iscclient -hostname ipserver -port 10999 -nodosconnecterr MYPROG
```

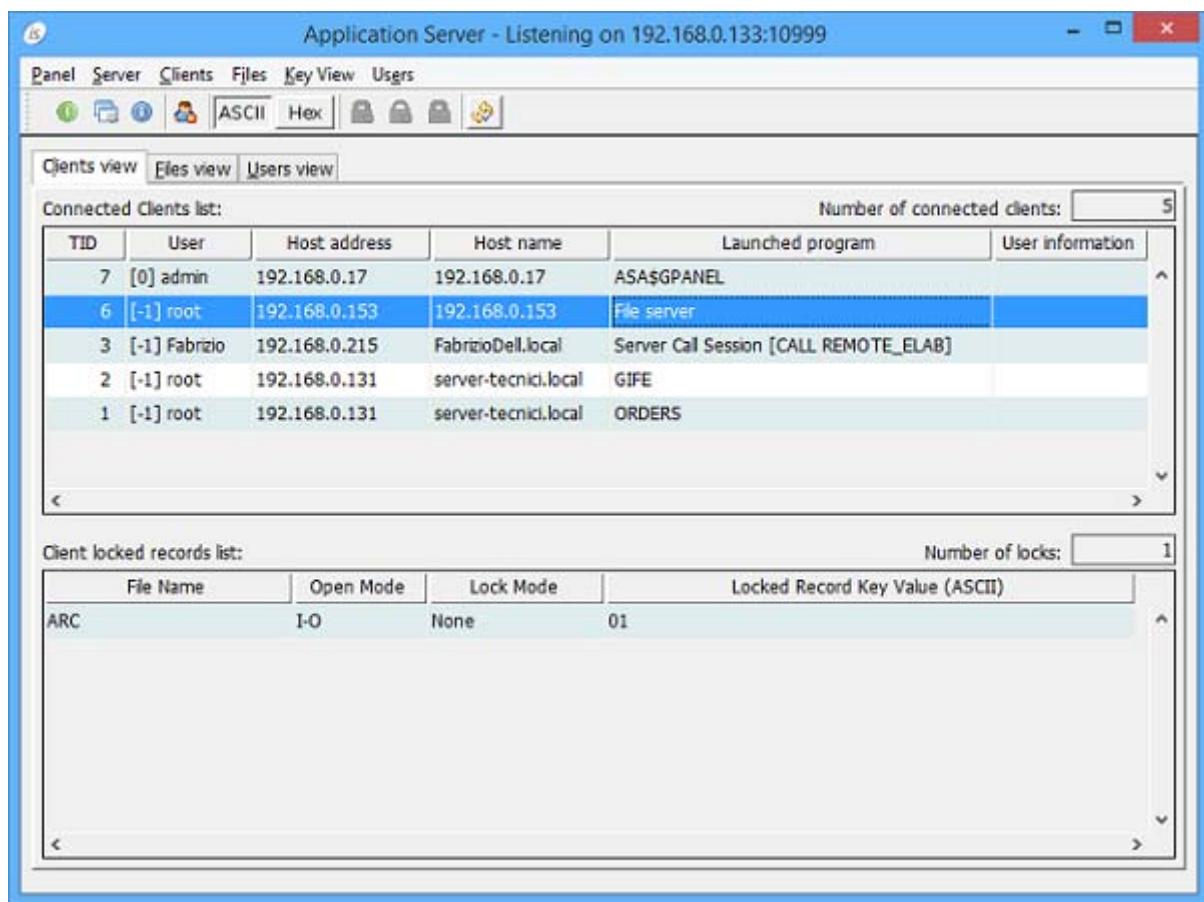
If no option is passed, by default the Connection lost error is returned as follows:



## Monitor users and locks of File Server

This new feature allows to monitor active users and locks in the File Server through the AS Panel. This allows the administrator user to have a complete list of connected users and locks in case iscobel.file.lock\_manager is set, like shown in the picture below.

This same info can be retrieved from cobol programs using the existing routines: A\$LIST\_USERS and A\$LIST\_LOCKS.



## Remote calls in –cp mode

The remote calls now support a new protocol named iscp, to allow COBOL programs, that use the C memory model (-cp compile option) to be called in remote way.

Example:

```
iscobol.remote.code_prefix=iscp://192.168.0.101:10999
```

It's also possible to specify multiple remote setting to define different directory or different server where call remotely COBOL program.

Example:

```
iscobol.remote.code_prefix=iscp://server1:10999\iscp://server2:10999
```

## isCOBOL DataBase Bridge Improvements

isCOBOL Database Bridge, the tool that enables COBOL programs to use power of RDBMS without changing COBOL source code or learning ESQL was enhanced to support a new database. This increase the number of databases supported natively.

## New IBM Informix support

The new option -di allows generating EDBI routines for Informix. All EDBI routines are optimized to generate specific SQL supported from IBM Informix DBMS.

## Improved performance

The EDBI routines now use a faster method to execute the corresponding SQL code of READ KEY from COBOL on Microsoft SQL Server. The EDBI RDBMS subroutines will benefit from this increase of performance up to 30%.

## Multi-connection support for COBOL threads

Usually COBOL threads shared DBMS connections unless the following property is used:

```
iscobol.jdbc.thread_connection=true
```

With this property turned on, each COBOL threads runs in a separate DBMS connection.

## isCOBOL EIS improvements

### Configuration to activate stateless mode

A COBOL Services can be designed to be stateless or statefull according with COBOL developers needs. In case of migration from CGI COBOL program that are stateless by design, this could be useful to force isCOBOL service to run as stateless service even if they was not designed to be stateless.

A new property iscobol.http.stateless has been added to force service to run as stateless.

Configuration examples:

```
iscobol.http.stateless=true
```

### webDirect enhancements

isCOBOL webDirect now includes ZK updated to version 8. In addition to take advantage of new ZK features, there are new configuration properties:

- iscobol.wd2.additional\_stylesheet=filename to load an additional CSS stylesheet
- iscobol.wd2.style=bs to use Bootstrap styling for controls

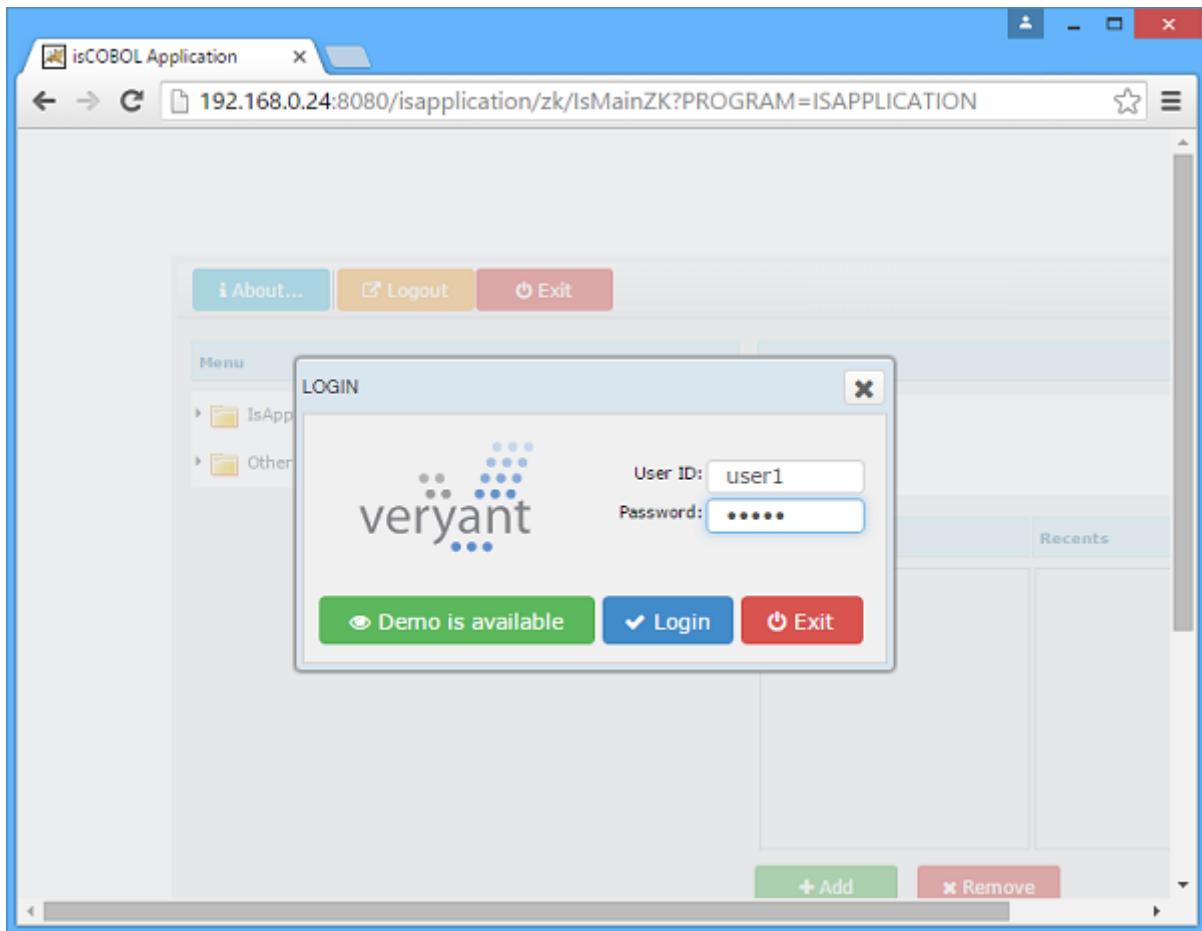
and new properties in the GUI syntax supported by isCOBOL compiler that will be used when running under webDirect:

- css-base-style-name (available for all controls) to set a css-style allowing a completely customized
- css-icon (available for push-button) to use font-awesome icons

Code example:

```
03 PB-LIST
Push-Button
title "&Login"
...
css-base-style-name "btn btn-primary"
css-icon "fa-check".
```

The result of this button is shown in the picture below:



TAB-CONTROL with ACCORDION style is supported also under webDirect. This is useful on web applications to have nice menu as alternative of tree-view control.

The image shown below depicts the final result with the ACCORDION style set on TAB-CONTROL in a browser.

Page1												
Page2												
Songs												
<b>This is the third page</b>												
<table border="1"><tr><td>Foo Foo</td><td>Santana</td><td></td></tr><tr><td>Help!</td><td>Beatles</td><td></td></tr><tr><td>Let It Be</td><td>Beatles</td><td></td></tr><tr><td>Lightning in the sky</td><td>Santana</td><td></td></tr></table>	Foo Foo	Santana		Help!	Beatles		Let It Be	Beatles		Lightning in the sky	Santana	
Foo Foo	Santana											
Help!	Beatles											
Let It Be	Beatles											
Lightning in the sky	Santana											
Last Page												

**Exit**

## isCOBOL 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

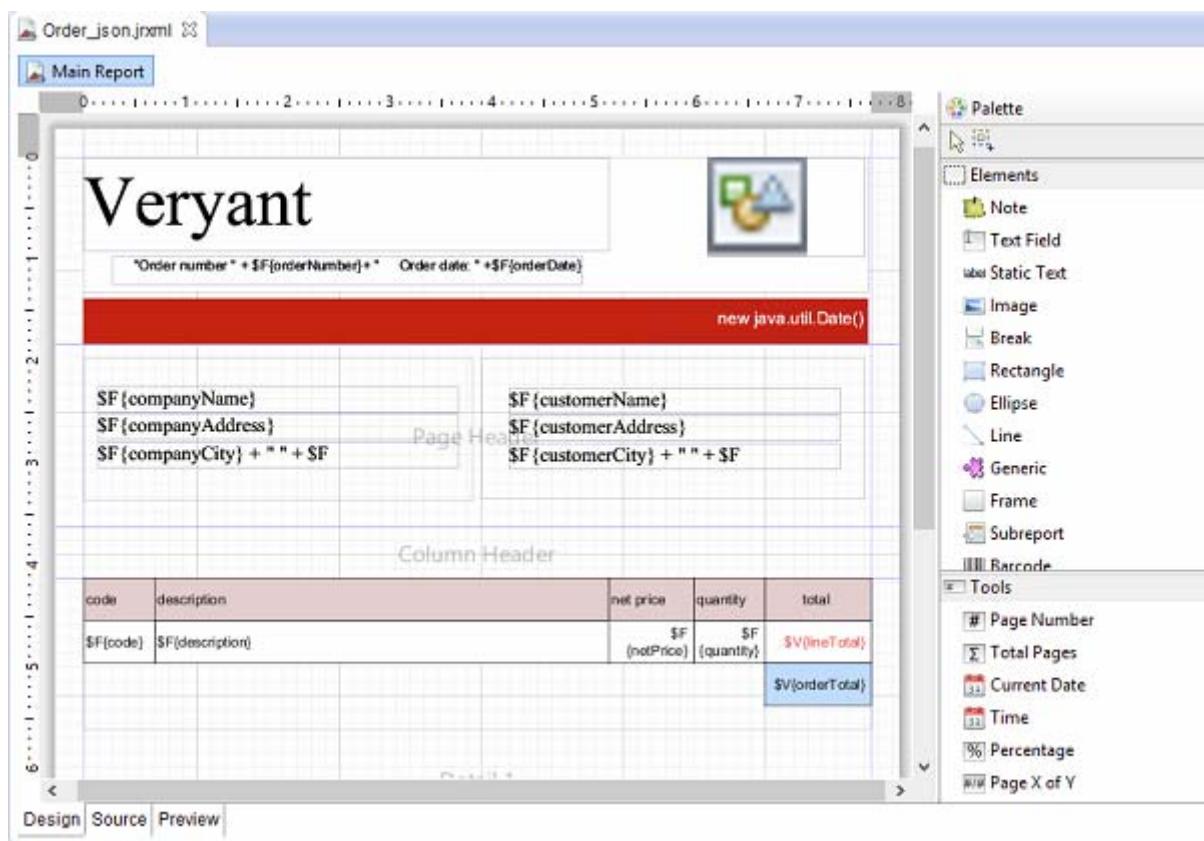
In isCOBOL Evolve 2015 R1, the isCOBOL IDE has been created with two different editions:

- 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

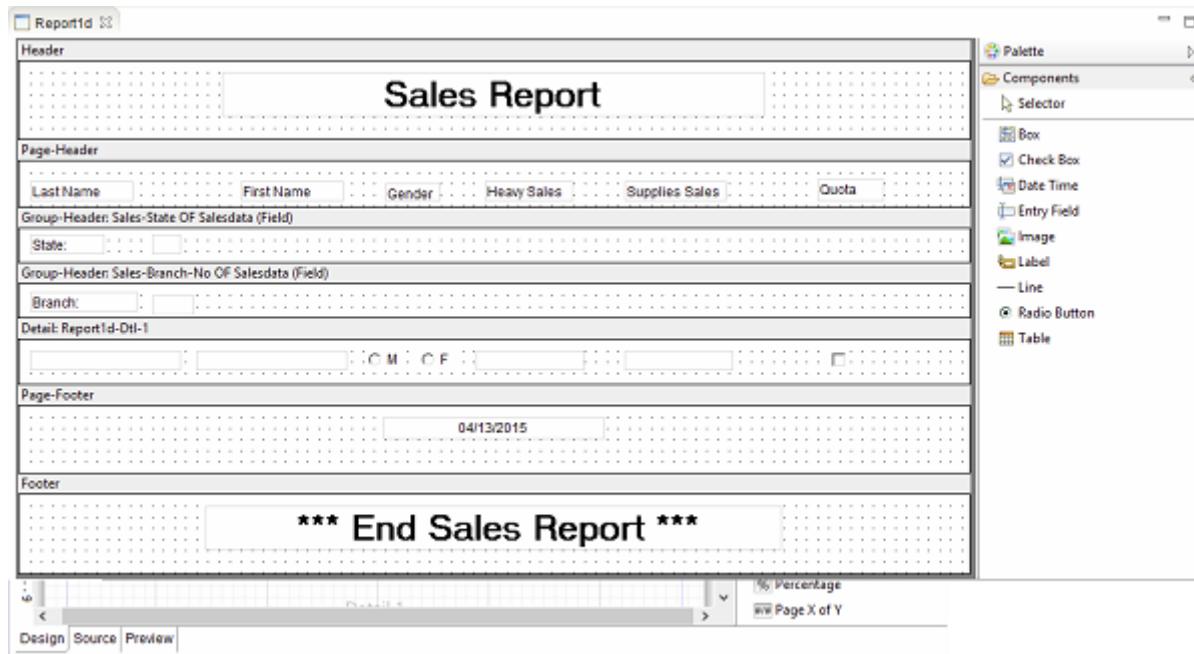
The addition of Jasper report studio to the isCOBOL IDE, as shown in the picture below (Jasper Studio Designer), allows isCOBOL users to create sophisticated layouts containing charts, images, sub-reports, crosstabs and much more. Moreover it enables you to access your data through JDBC, TableModels,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



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 the picture below, 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.



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, it is also possible to continue using acubenchprint.dll if needed.

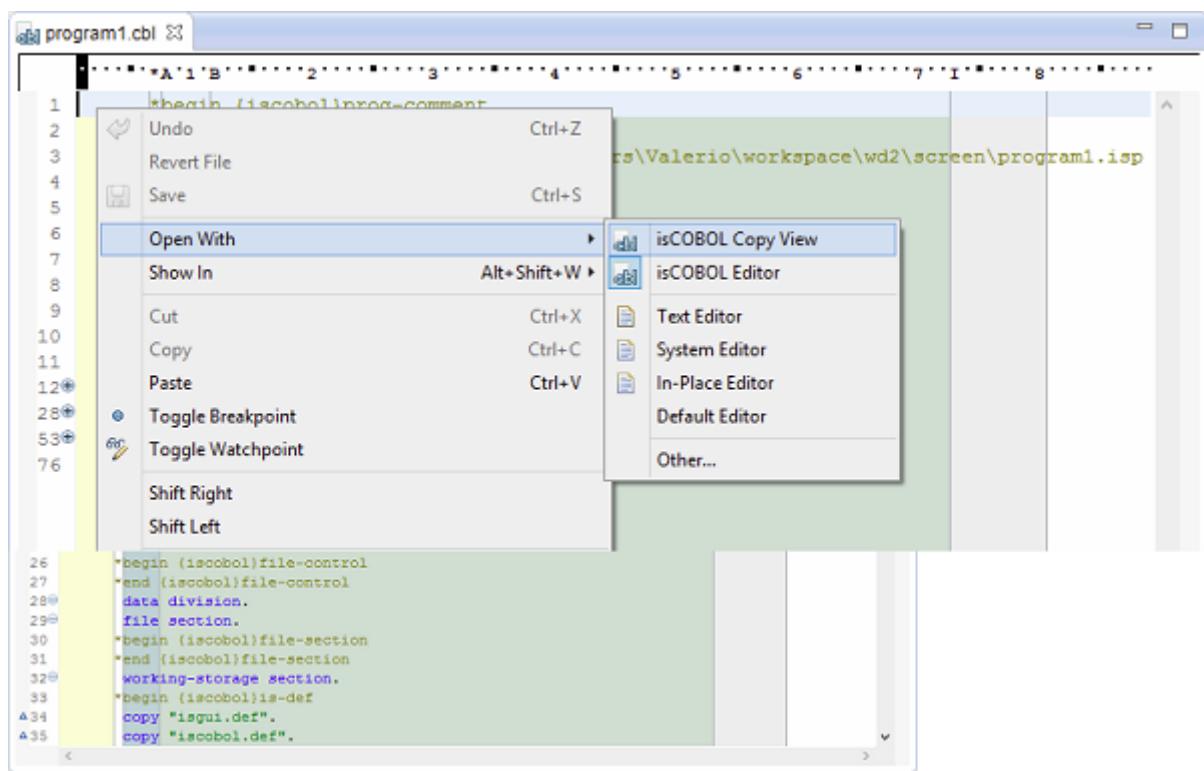
- Enhanced code folding

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.

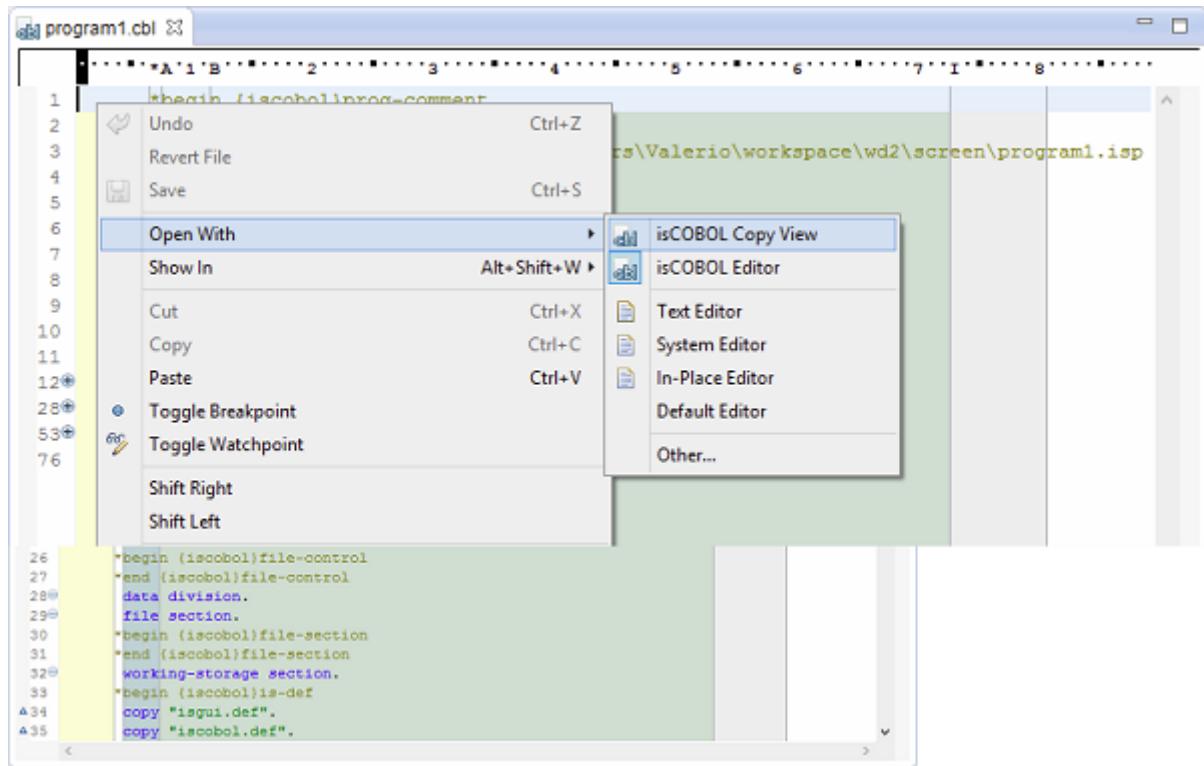
```
18      *end {iscobol}alphabet
19      *begin {iscobol}decimal-point
20      *end {iscobol}decimal-point
21      repository.
22      *begin {iscobol}repository
23      *end {iscobol}repository
24      input-output section.
25      file-control.
26      *begin {iscobol}file-control
27      *end {iscobol}file-control
28      data division.
29      file section.
30      *begin {iscobol}file-section
31      *end {iscobol}file-section
32      working-storage section.
33      *begin {iscobol}is-def
34      copy "isgui.def".
35      copy "iscobol.def".
```

- 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 the picture below, is a new entry on existent “Open With” menu.



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 the picture below, new isCOBOL View including all copy files.



- 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.

Example command for PSF:

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

Example command for DLT:

```
isIDE.exe -data C:\MyWorkspace -nosplash -application
com.iscobel.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 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 file-status 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;

-cpf36 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.

-cpf36 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 supported syntax

ACCEPT FROM WINDOW allow now to specify OF THREAD H-THREAD to receive the window handle of specified thread. This allow to easily retrieve from a different thread running (for example the main menu of the application), the window handle of a separated thread (for example a called program in thread mode) to set the input window as required.

Example:

```
accept hwin-pgm from window of Thread22
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 iscobel.compiler.iss\_julian\_base was providing 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:

```
iscobel.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

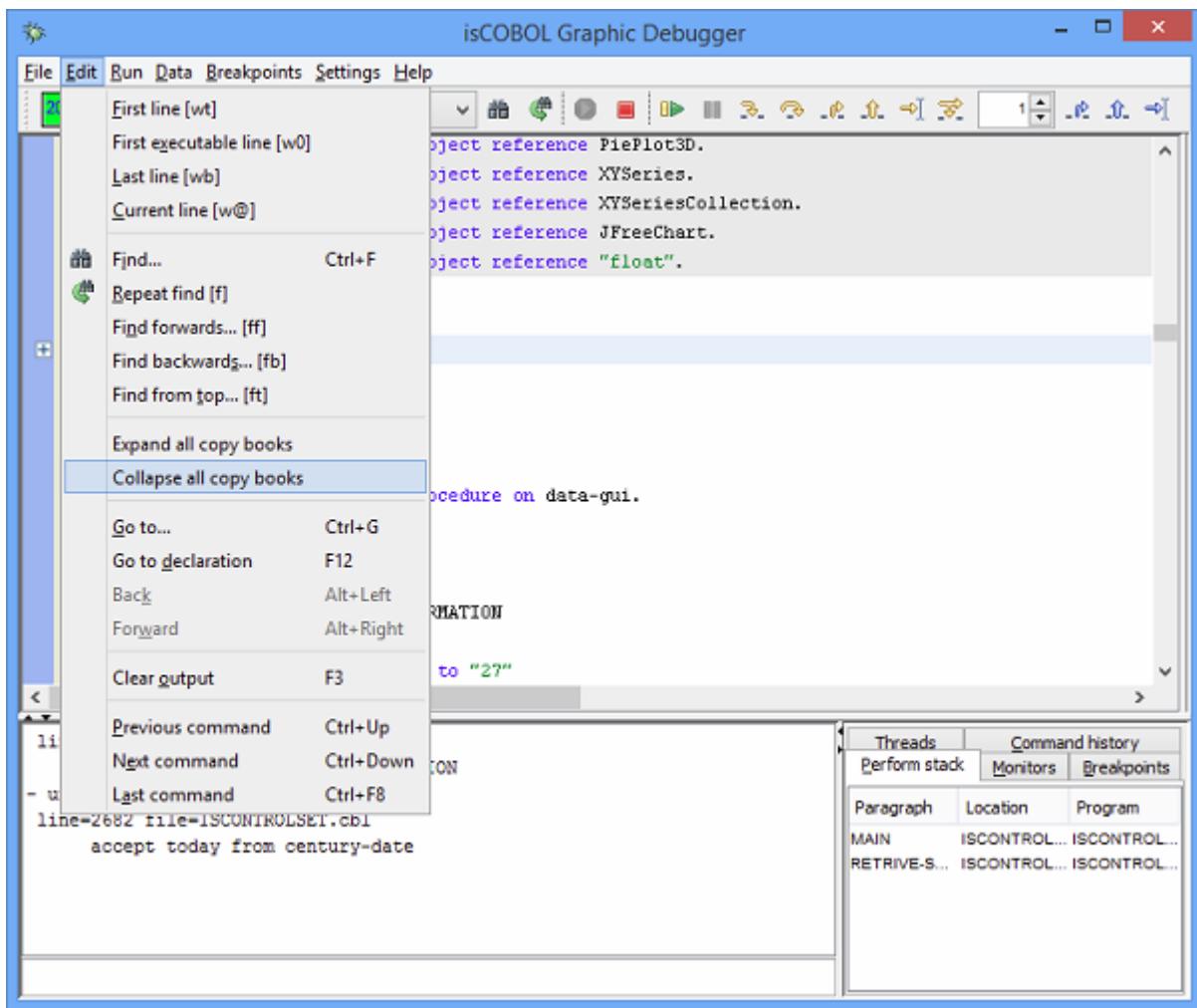
Now the isCOBOL Debugger allows you to expand copy files to see all the COBOL code. As depicted in the figure below, a “plus” symbol allows you to expand or collapse related copy files.

The screenshot shows the isCOBOL Graphic Debugger interface. The main window displays the COBOL source code for 'ISCONTROLSET.cbl'. The code includes various declarations, procedures, and system calls. A tooltip 'Expand copy' is visible over a line of code. Below the code editor is a command history pane showing recent commands entered. To the right, there is a panel for monitoring threads and breakpoints.

```
File Edit Run Data Breakpoints Settings Help
20.503/126.877Mb ISCONTROLSET.cbl
7 77 wPiePlot3D          object reference PiePlot3D.
8 77 wXYSeries           object reference XYSeries.
9 77 wXYSeriesCollection object reference XYSeriesCollection.
10 77 wJFChart             object reference JFreeChart.
11 77 tmpFloat             object reference "float".
46
47  screen section.
48  Expand copy
49
50 procedure division.
51 declaratives.
52 SET-GRID section.
53 use after standard error procedure on data-gui.
54 continue.
55 end declaratives.
56 MAIN.
57 perform RETRIEVE-SYSTEM- INFORMATION
58
59 set environment "quit-mode" to "27"
line=57 file=ISCONTROLSET.cbl
    perform RETRIEVE-SYSTEM- INFORMATION
- unknown command 'sort-gri'
line=2682 file=ISCONTROLSET.cbl
    accept today from century-date

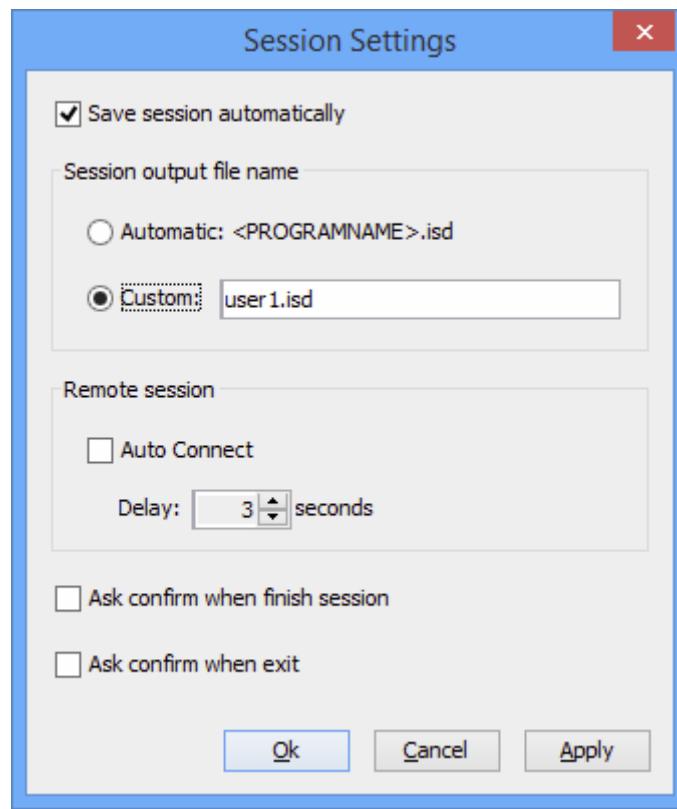
Threads Command history
Perform stack Monitors Breakpoints
Paragraph Location Program
MAIN ISCONTROL... ISCONTROL...
RETRIEVE-S... ISCONTROL... ISCONTROL...
```

You can also expand or collapse all copy files of the whole COBOL source that are going to be debugged.



- 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 the picture below, a debugger user can customize the name of debugger session file.



- 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, iteratively:

The screenshot shows the isCOBOL Graphic Debugger interface. The main window displays a COBOL source code file named ISCONTROLSET.cbl. The code includes sections for special-names, input-output, file, and record descriptions. A tooltip is visible over the 'DATA-GUI' variable. The bottom-left pane shows the command history and stack trace, indicating a breakpoint at line 2066. The bottom-right pane shows the current paragraph, location, and program details.

```
8 special-names.
9 decimal-point is comma.
10
11 input-output section.
12 file-control DATA-GUI = Opened INPUT last file-status = '00'
13   select data-gui assign to path-data-gui
14     class "com.iscobol.io.RemoteRelative"
15     status file-status.
16
17 file section.
18 fd data-gui.
19 01 rec-data-gui.
20   03 bk-page      pic 99.
21   03 bk-displ    pic x(40).
22   03 bk-width    pic 9(4).
23   03 bk-vpadd   pic 9(3).
24   03 bk-ordin    pic x(40).
25   03 bk-sort     pic x(60).
26   03 bk-sclstd   pic 9(5).
27   03 bk-sccstd   pic 9(5).

line=57 file=ISCONTROLSET.cbl
  perform RETRIEVE-SYSTEM- INFORMATION
+ hit breakpoint at line 2066, file ISCONTROLSET.cbl
line=2066 file=ISCONTROLSET.cbl
  open input data-gui
line=2067 file=ISCONTROLSET.cbl
  if file-status = "00"

Threads Command history
Perform stack Monitors Breakpoints
Paragraph Location Program
MAIN ISCONTROL... ISCONTROL...
LOAD-DATA... ISCONTROL... ISCONTROL...
```

Two additional items are available in the popup menu of the output window in the Debugger to easily maintain it clean and copy the output:

The screenshot shows the isCOBOL Graphic Debugger interface. The main window displays COBOL source code in a syntax-highlighted editor. A context menu is open over line 2697, listing options: Clear, Copy, and Select All. The menu is displayed in a light gray background with black text. The source code includes several calls to environment variables like J\$NETADDRESS and C\$GETENV.

```

2681 RETRIEVE-SYSTEM-INFORMATION.
2682 accept today from century-date
2683 accept wtime from time
2684 accept system-information from system-info
2685 accept terminal-abilities from terminal-info.
2686
2687 perform RETRIEVE-SERVER-INFORMATION
2688
2689 if is-remote
2690   perform RETRIEVE-CLIENT-INFORMATION
2691 end-if.
2692
2693 RETRIEVE-SERVER-INFORMATION.
2694
2695 call "J$NETADDRESS" using s-hostname s-hostip
2696
2697 call "C$GETENV" using "java.vm.name"
2698           s-java-vm-name
2699
2700 call "C$GETENV" using "java.version"

```

## 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 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		0XXXXXXXXXX			ZZZZZZZZZZZ
2		1XXXXXXXXXX	r1	r2	ZZZZZZZZZZZ
3		0YYYYYYYYYY	r1	r2	ZZZZZZZZZZZ
4		1YYYYYYYYYY	r11	r22	ZZZZZZZZZZZ

Now they're filled as follows:

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

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 iscobel.easydb.replacement\_rules

The new configuration property iscobel.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.

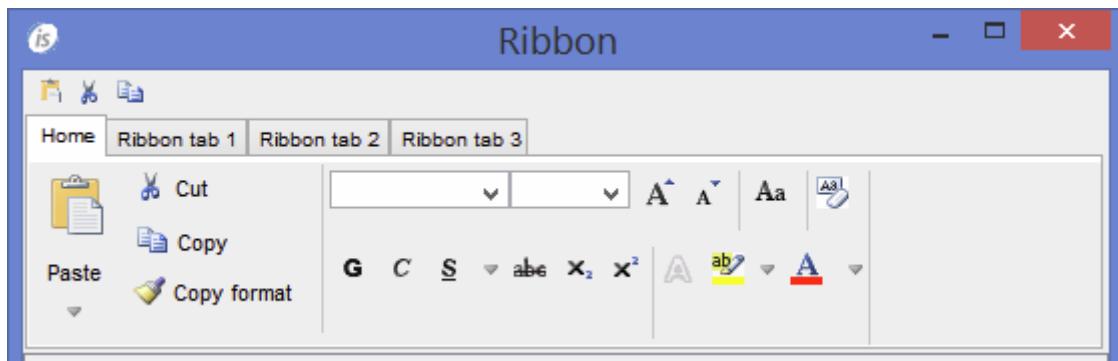
Such ribbons use tabs to expose different sets of controls, eliminating the need for many parallel toolbars and menu bars.

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.

The following picture shows the final result with the ON-HEADER style set on the buttons Paste, Cut and Copy.



- 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 ALLOWCONTAINER, 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-brower

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:

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

That creates an iscobel.hprof.txt file containing:

isCOBOL profile rev \$Revision: 18624 \$ created Tue Apr 14 14:35:08 CEST 2015				
elapsed time	evaluated time	overhead1=56; overhead2=75	self %	accum %
seconds	count		program:paragraph	
25.96%	0.65	1	IO_INDEXED:DELETE_FILE1_TEST	
16.09%	0.40	1	IO_INDEXED:LOAD_FILE1_TEST	
14.80%	0.37	1	IO_INDEXED:UPDATE_FILE1_TEST	
9.15%	0.23	1	IO_INDEXED:READ_FILE1_TEST	
8.66%	0.21	1	IO_SEQUENTIAL:LOAD_FILE1_TEST	
5.34%	0.13	1	IO_RELATIVE:UPDATE_FILE1_TEST	
4.56%	0.11	1	IO_RELATIVE:DELETE_FILE1_TEST	
3.22%	0.08	1	IO_PERFORMANCE:MAIN_LOGIC	
3.20%	0.08	1	IO_RELATIVE:LOAD_FILE1_TEST	
3.12%	0.07	1	IO_SEQUENTIAL:READ_FILE1_TEST	
2.73%	0.06	1	IO_RELATIVE:READ_FILE1_TEST	
1.69%	0.04	1	IO_INDEXED:MAIN_LOGIC	
0.52%	0.01	4	IO_INDEXED:START_TIMER	
0.40%	0.01	4	IO_RELATIVE:START_TIMER	
0.37%	0.00	2	IO_SEQUENTIAL:START_TIMER	
0.03%	0.00	1	IO_RELATIVE:MAIN_LOGIC	
0.03%	0.00	1	IO_SEQUENTIAL:MAIN_LOGIC	
0.01%	0.00	4	IO_INDEXED:STOP_TIMER	
0.01%	0.00	4	IO_RELATIVE:STOP_TIMER	
0.00%	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-SETBACKGROUND-COLOR was provided. It sets the background-color to be used during printing.

Code example:

```
initialize wpidata-text-color
move wppt-color-yellow to wpidata-text-color
call "win$printer" using WINPRINT-SET-BACKGROUND-COLOR
wpidata-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 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 set dynamically the fetch size for a ESQL cursor statement.

A new property named iscobol.jdbc.fetch\_size allows specifying the number of rows to retrieve in one go from the database. This can be used in in ESQL programs or with isCOBOL Database Bridge.

- I\$IO info-function now 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 be able to specify DLL calling convention during CALL statement.

To force all function of mylib wil be called with C convention:

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

To force all function of mylib wil be called 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 uses of memory.

- Encrypted SSL connections

SSL is an industry standard and is used by millions of websites in the protection of their online transactions with their customers. Users that need that all data passed between the isCOBOL Server and isCOBOL Thin Client remain private and integral can take advantage of this SSL standard support.

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

On the server side:

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

On the client side:

```
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 packet used. Starting from isCOBOL 2015R1, many areas were improved in the Application Server traffic in order to increasing screen performances.

For example on DISPLAY screen-section and ACCEPT statements we used xx% less number of packets. Also INQUIRE statement, that usually break any optimizer option forcing 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 setting. This optimization reducing the time spent to the isCOBOL Thin client to go live.

## isCOBOL EIS improvements

- Simplified the migration from legacy CGI programs

The IS EXTERNAL-FORM clause 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 contains substitution marker can be merge 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 precompiler 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

Enhanced HTTPHandler class adding the ability to process html string with the new method:

```
processHtmlString(htmlString)
```

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 a external file.

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.
...
lnk-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 is analogous to doPost, however it send the parameters using the multipart/form-data protocol.

The second method allows to save 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)
```

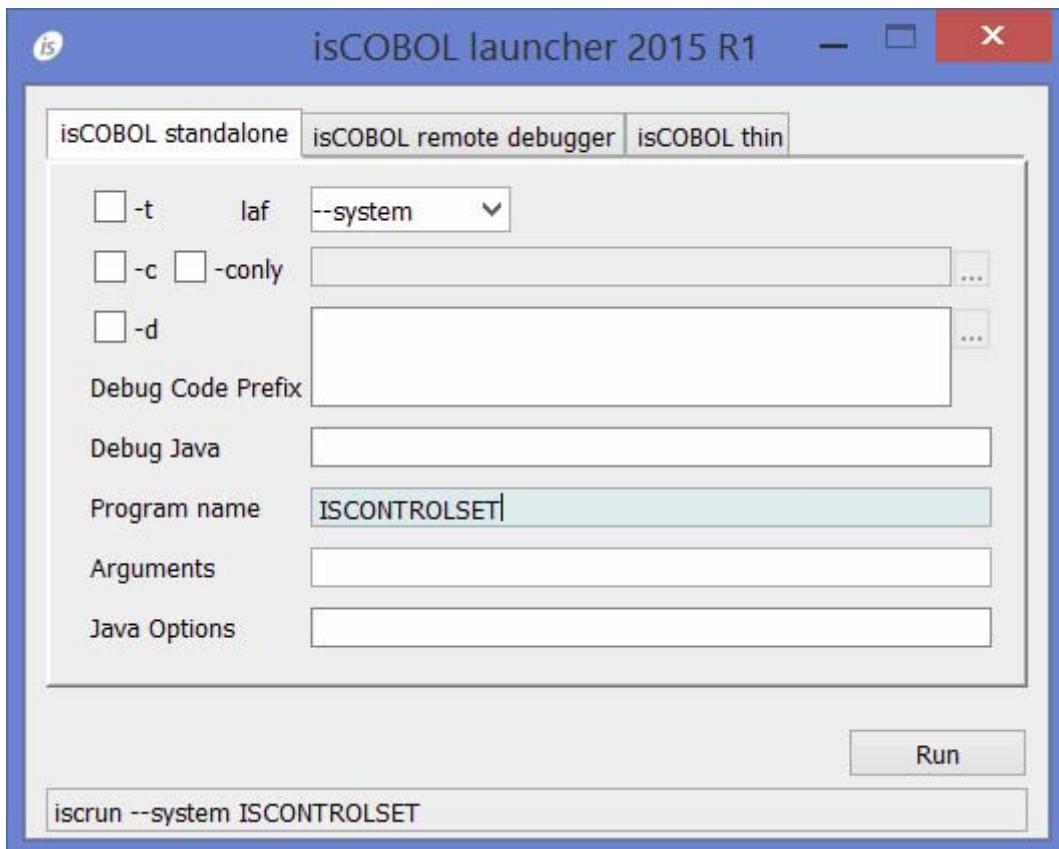
These methods allow to use the previous doPostMultipart method in HttpClient class. The latter set the mime type of the file as "application/octet-stream". Example:

```
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 the screenshot below, 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.



- New GUI interface for XML2WRK utility

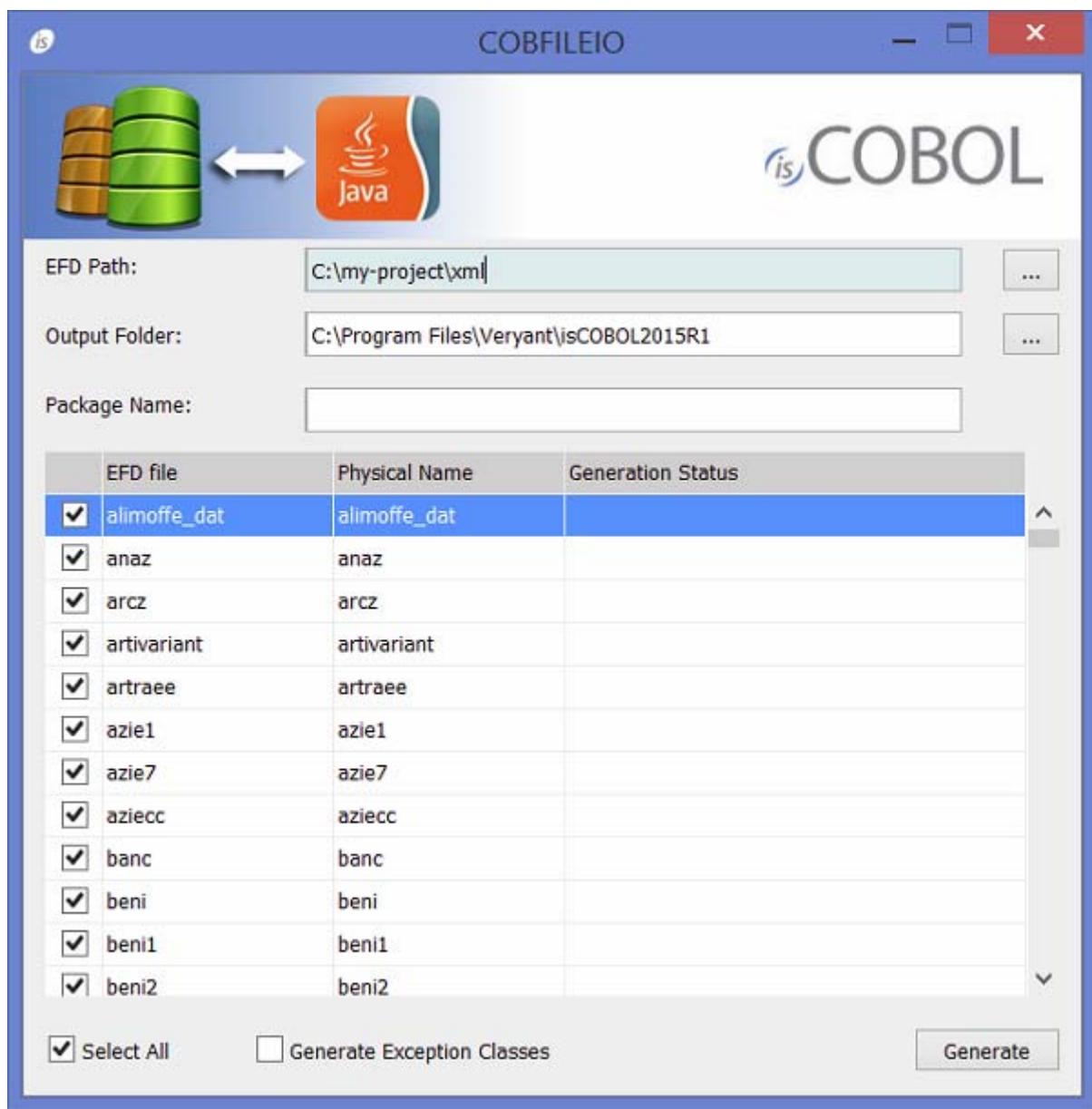
The XML2WRK utility that opens an XML file and creates the corresponding record definition used with the XMLStream Class (com.iscobol.rts.XMLStream) object, is now released with a graphical user interface improving its usage. As depicted in the picture below, in addition to the new GUI interface, it is also possible to parse remote xml files via URL and generates namespaces.



- New GUI interface for COBFILEIO utility

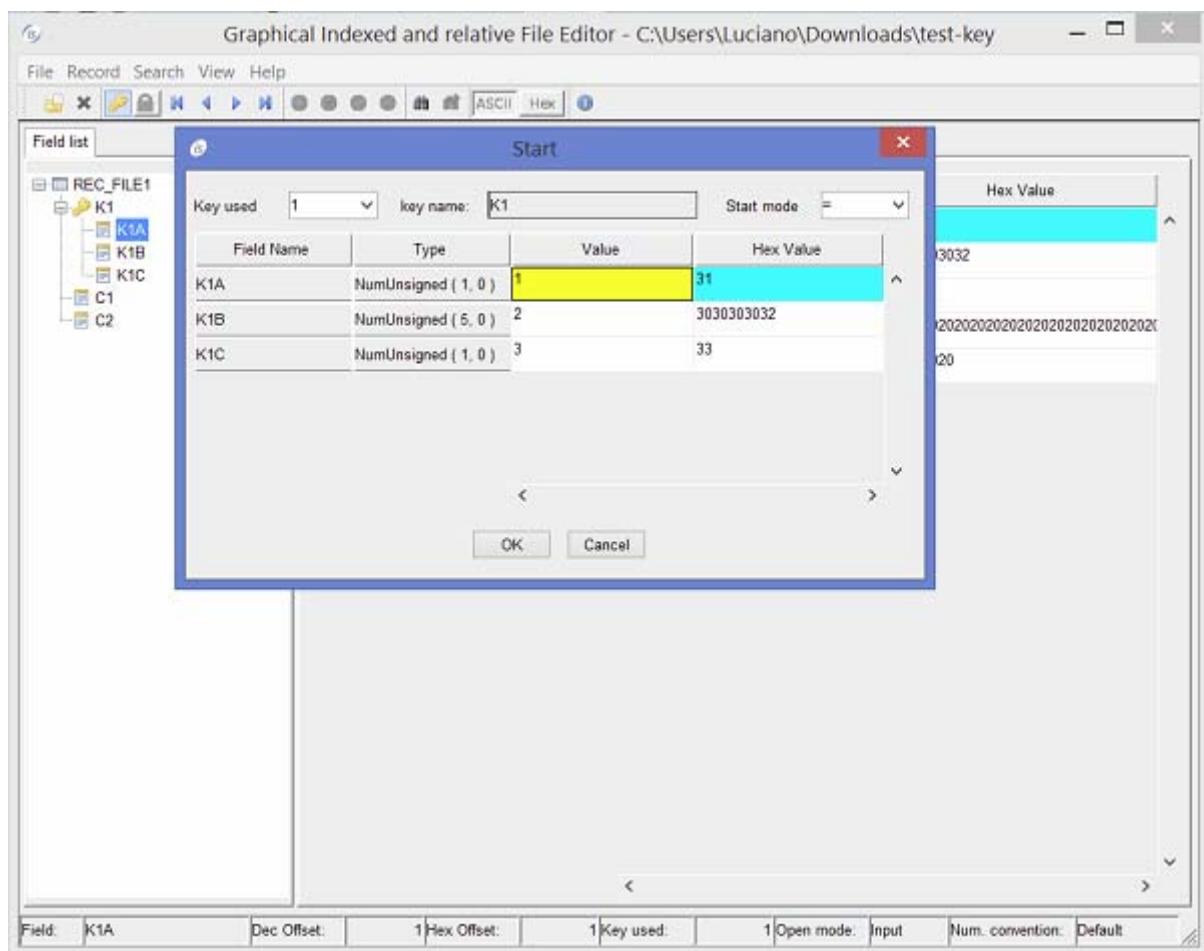
COBFILEIO, the utility that generates Java classes used to access COBOL ISAM files and records from Java programmer, is now released with a graphical user interface. As depicted in the picture below, this new GUI will make easier to be used for all users.

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



- 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.



- ISMIGRATE improvements

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

