isCOBOL^{TM} Evolve

isCOBOL Evolve 2023 Release 1 Overview

isCOBOL Evolve 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.

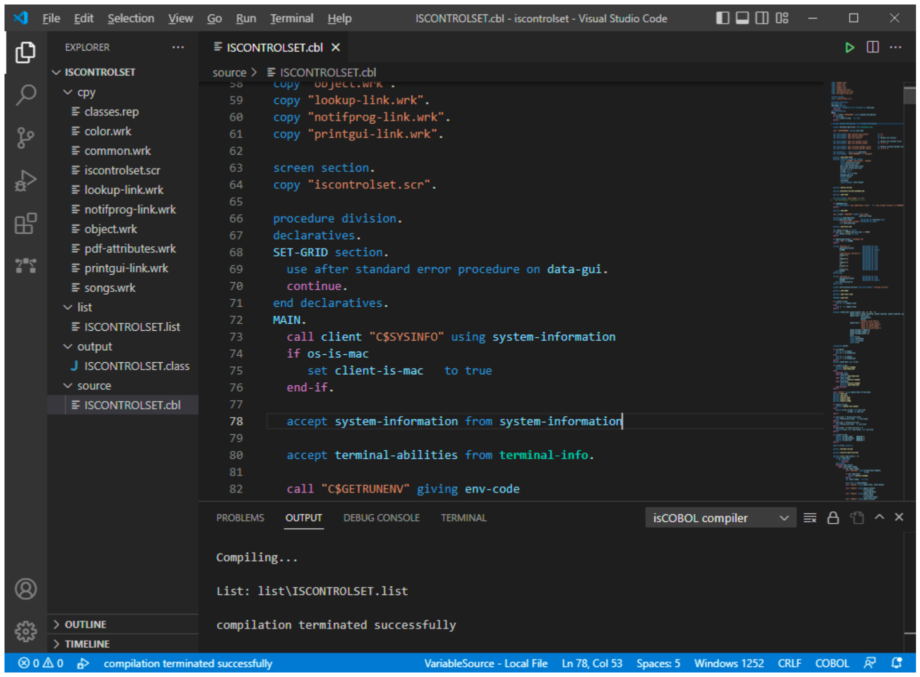
The Veryant isCOBOL Extension provides several useful new commands, accessible using the shortcut key Shift-Ctrl-P on Windows and Linux and Shift-Cmd-P on MacOS.   COBOL developers can use these commands to create new isCOBOL projects, create new source and copybook files with a default template and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



```
copy "object.wrk".
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-information

  accept terminal-abilities from terminal-info.

  call "C$GETRUNENV" giving env-code
```

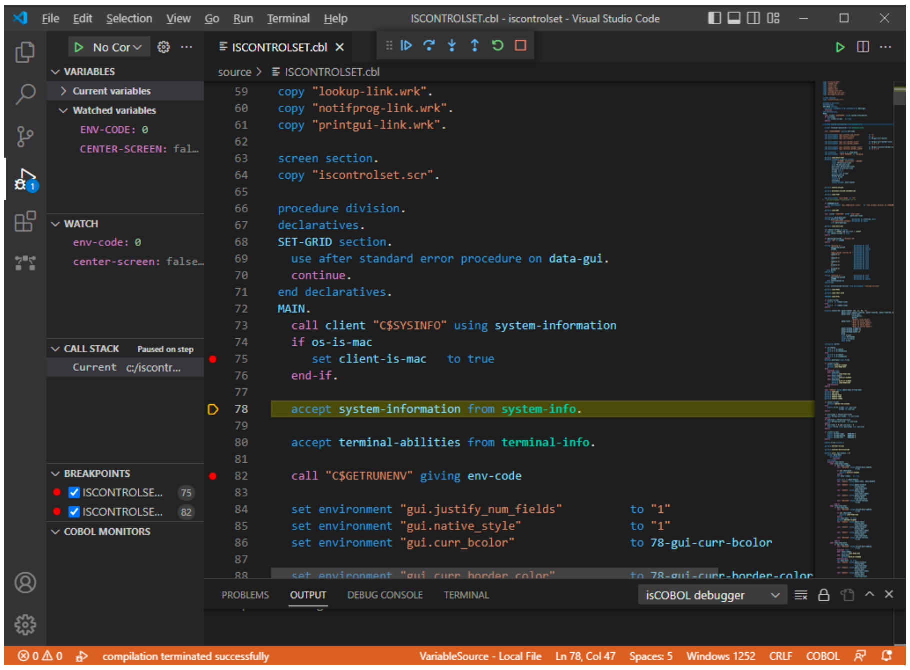
```
Compiling...

List: list\ISCONTROLSET.list

compilation terminated successfully
```

⊗ 0 ⚠ 0 compilation terminated successfully VariableSource - Local File Ln 78, Col 53 Spaces: 5 Windows 1252 CRLF COBOL

Figure 2. Debugging in the Veryant isCOBOL Extension.



```
ENV-CODE: 0
CENTER-SCREEN: fal…
```

```
env-code: 0
center-screen: false…
```

```
Current   c:/iscontr…
```

```
ISCONTROLSE…  75
ISCONTROLSE…  82
```

```
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.

  accept terminal-abilities from terminal-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 78-gui-curr-bcolor

  set environment "gui.curr_border_color"           to 78-gui-curr-border-color
```

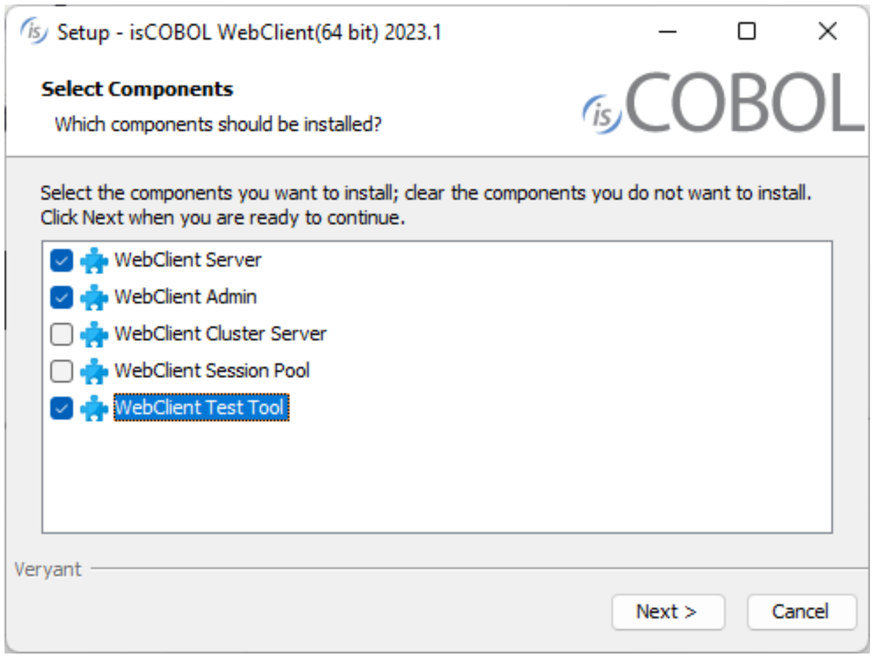
⊗ 0 ⚠ 0 compilation terminated successfully VariableSource - Local File Ln 78, Col 47 Spaces: 5 Windows 1252 CRLF COBOL

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.



Setup - isCOBOL WebClient(64 bit) 2023.1

Select Components

Which components should be installed?

Select the components you want to install; clear the components you do not want to install.
Click Next when you are ready to continue.

- [x] WebClient Server
- [x] WebClient Admin
- [ ] WebClient Cluster Server
- [ ] WebClient Session Pool
- [x] WebClient Test Tool

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:

```
webcclient-testtool
```

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

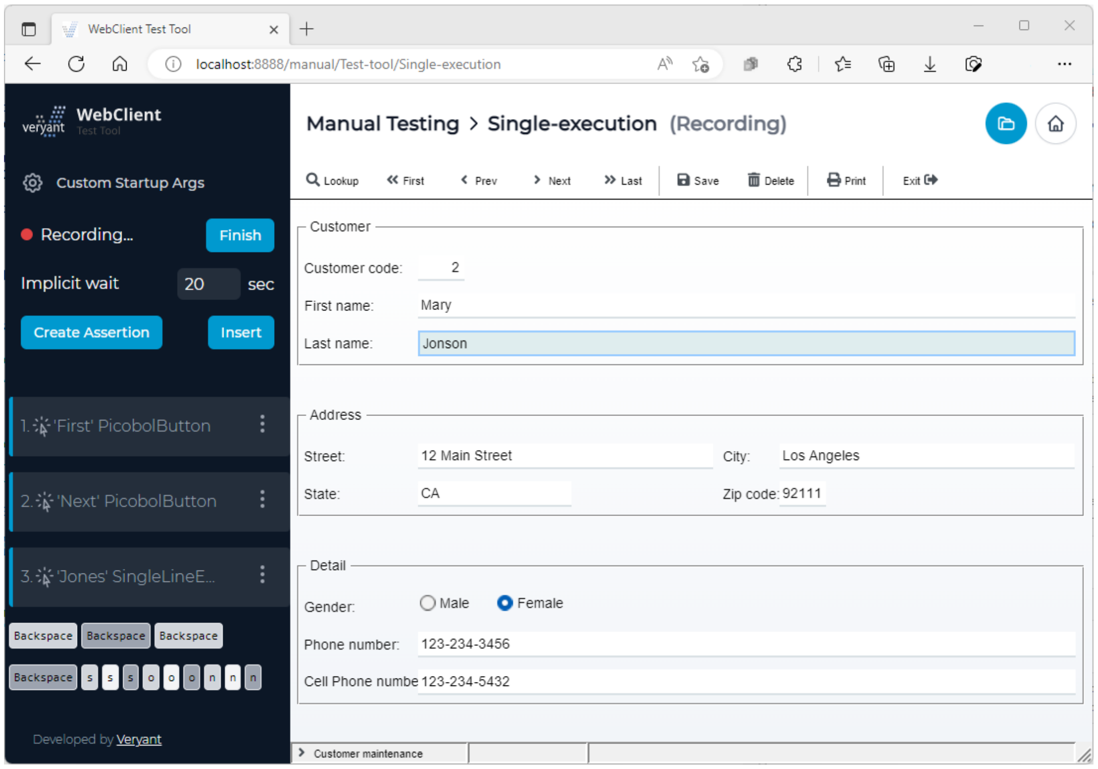
```
webclient-testtool -start
```

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.



Manual Testing > Single-execution (Recording)

Implicit wait: 20 sec

1. 'First' PicobolButton
2. 'Next' PicobolButton
3. 'Jones' SingleLineE...

Customer

Customer code:	2
First name:	Mary
Last name:	Jonson

Address

Street:	12 Main Street	City:	Los Angeles
State:	CA	Zip code:	92111

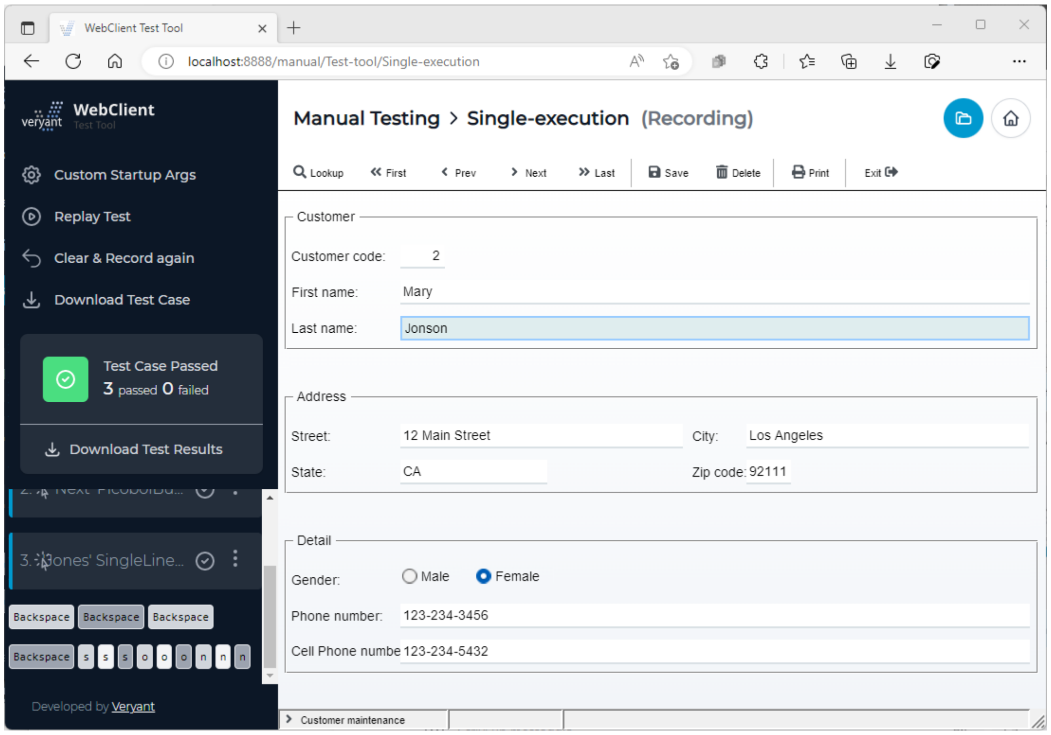
Detail

Gender:	Male [✓ Female]
Phone number:	123-234-3456
Cell Phone numbe	123-234-5432

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.



Manual Testing > Single-execution (Recording)

Test Case Passed
3 passed 0 failed

3. Jones' SingleLine...

Customer

Customer code:	2
First name:	Mary
Last name:	Jonson

Address

Street:	12 Main Street	City:	Los Angeles
State:	CA	Zip code:	92111

Detail

Gender:	Male [✓ Female]
Phone number:	123-234-3456
Cell Phone numbe	123-234-5432

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

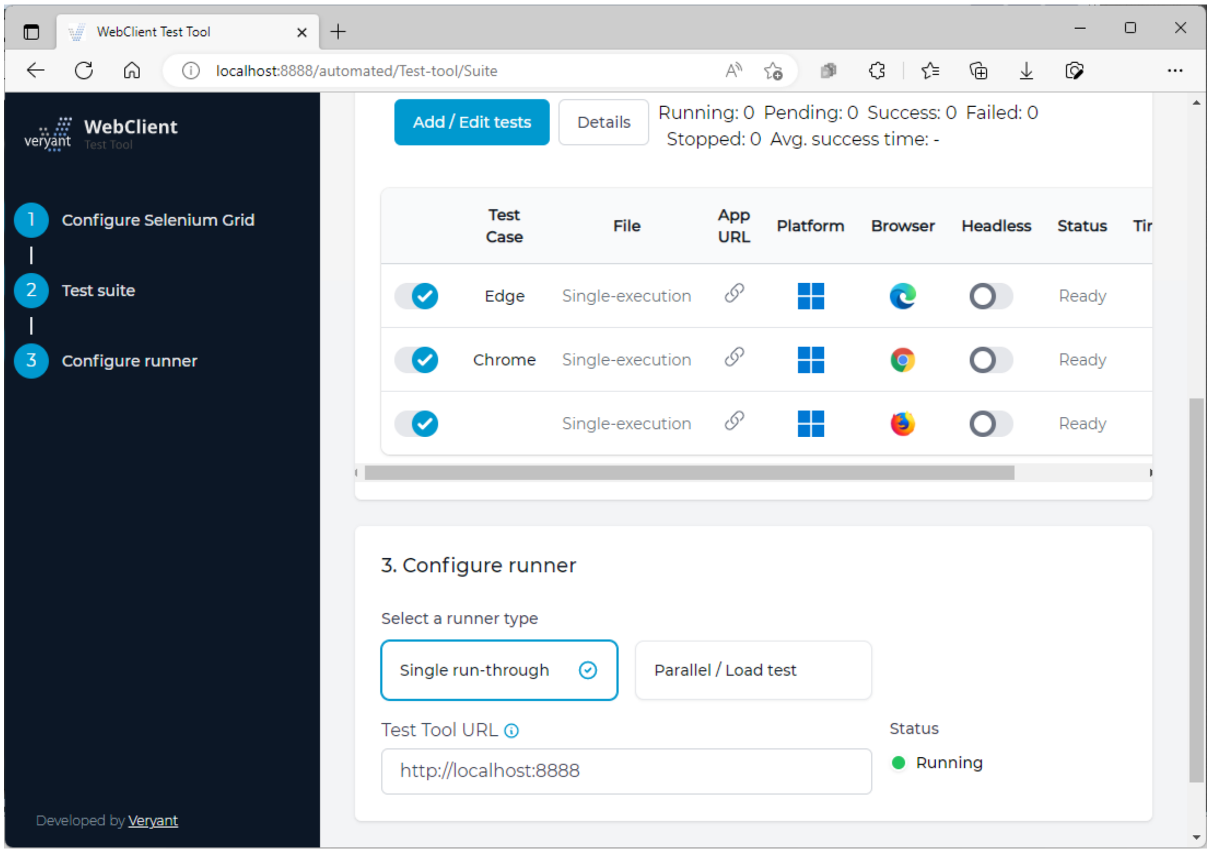
A suite is a configuration file where you define which test cases to run and set options to be used. Automated test cases require Selenium Grid, a smart proxy server that makes it easy to run tests in parallel on multiple machines. This is done by routing commands to remote web browser instances, where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



Running: 0 Pending: 0 Success: 0 Failed: 0
Stopped: 0 Avg. success time: -

	Test Case	File	App URL	Platform	Browser	Headless	Status
✓	Edge	Single-execution					Ready
✓	Chrome	Single-execution					Ready
✓		Single-execution					Ready

3. Configure runner

Select a runner type

Single run-through ✓ | Parallel / Load test

Test Tool URL: http://localhost:8888

Status: ● Running

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:

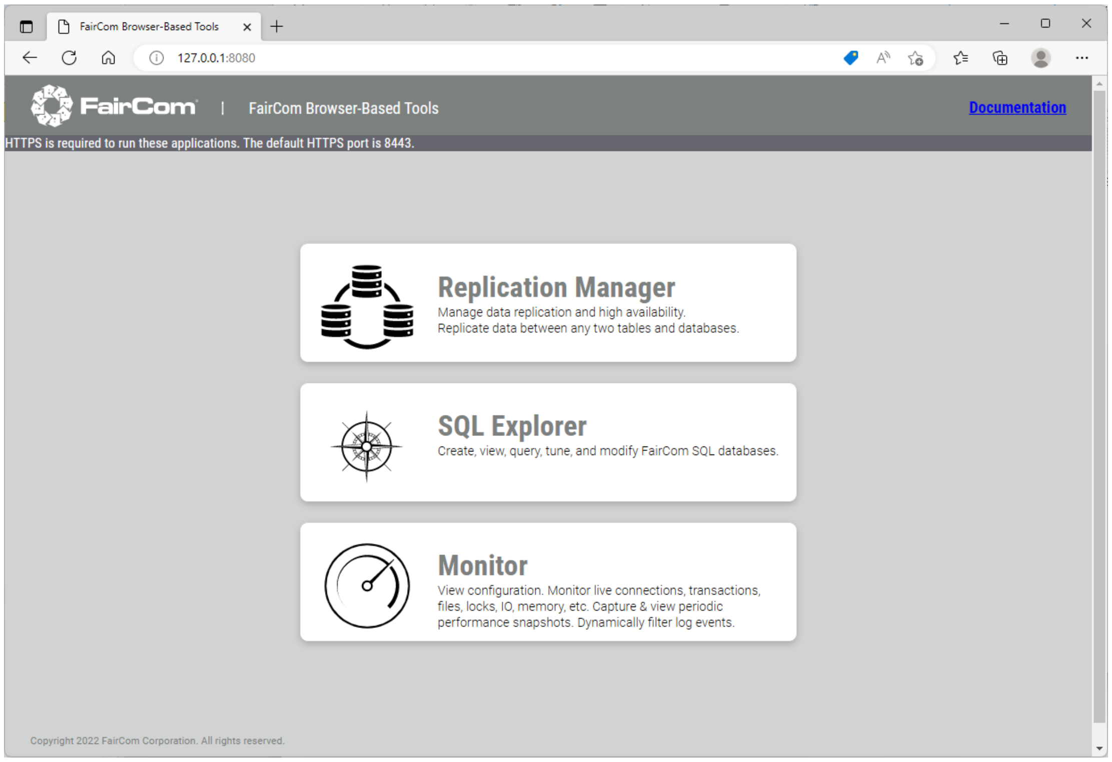
```
{
  "parallel": false,
  "maxRunningParallel": 1,
  "rampUpPeriod": 1,
  "runCount": 1,
  "testCaseParameters": [{
    "project": "project 1",
    "file": "test 1",
    "name": "Test",
    "webswingAppUrl": "http://localhost:8080/isapplication",
    "webswingUsername": "admin",
    "webswingPassword": "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.



HTTPS is required to run these applications. The default HTTPS port is 8443.

Replication Manager
Manage data replication and high availability.
Replicate data between any two tables and databases.

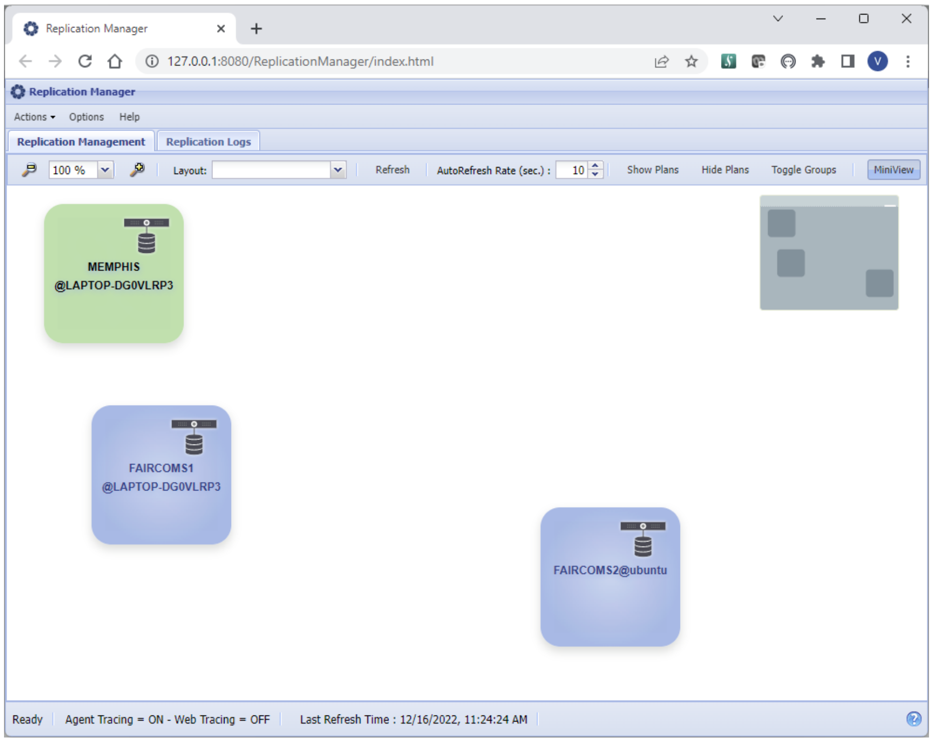
SQL Explorer
Create, view, query, tune, and modify FairCom SQL databases.

Monitor
View configuration. Monitor live connections, transactions, files, locks, IO, memory, etc. Capture & view periodic performance snapshots. Dynamically filter log events.

Copyright 2022 FairCom Corporation. All rights reserved.

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*.

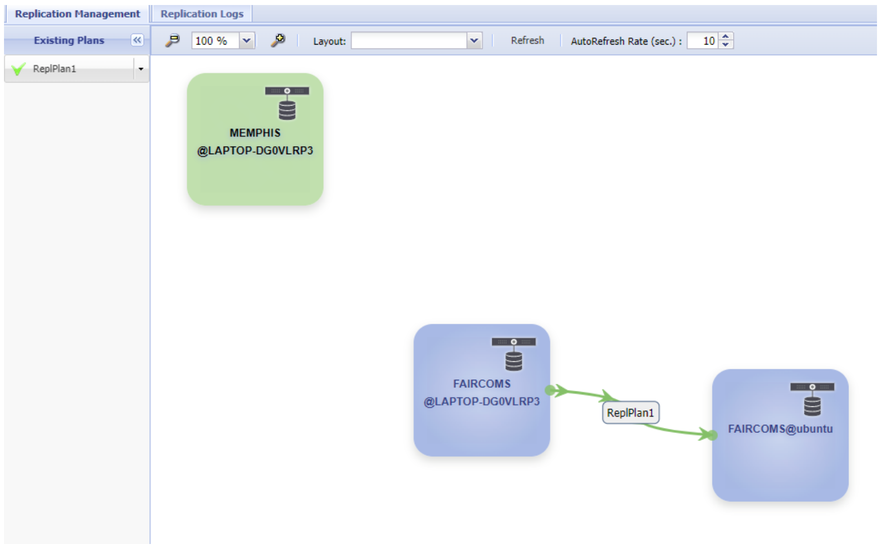
Figure 8. Replication Manager View.



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.

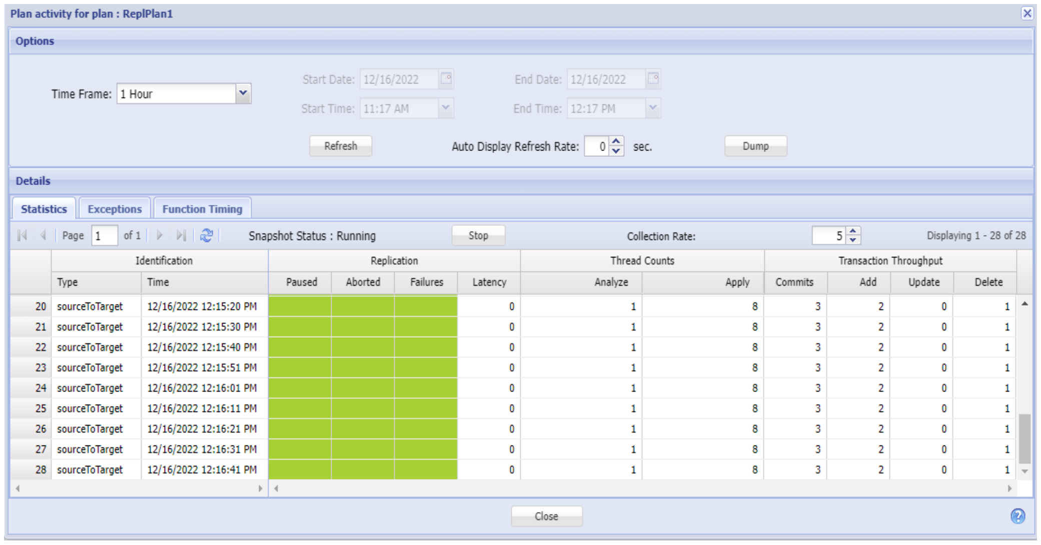
In the Replication Manager's application, you can drag a "Source" server over a "Target" server to create a Replication plan, based on a Publication where you can define if the replication is for SQL tables, ISAM files or both. You can also filter ISAM files included in specific folders that need to 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.



Time Frame: 1 Hour
Start Date: 12/16/2022
End Date: 12/16/2022
Start Time: 11:17 AM
End Time: 12:17 PM
Auto Display Refresh Rate: 0 sec.

Page 1 of 1 Snapshot Status : Running Collection Rate: 5 Displaying 1 - 28 of 28

	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	1	8	3	2	0	1
21	sourceToTarget	12/16/2022 12:15:30 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
22	sourceToTarget	12/16/2022 12:15:40 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
23	sourceToTarget	12/16/2022 12:15:51 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
24	sourceToTarget	12/16/2022 12:16:01 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
25	sourceToTarget	12/16/2022 12:16:11 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
26	sourceToTarget	12/16/2022 12:16:21 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
27	sourceToTarget	12/16/2022 12:16:31 PM	🟩	🟩	🟩	0	1	8	3	2	0	1
28	sourceToTarget	12/16/2022 12:16:41 PM	🟩	🟩	🟩	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. Using this statement CONNECT RESET 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

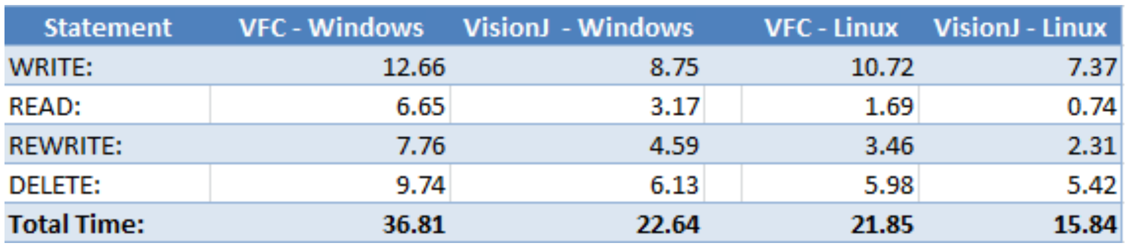
The isCOBOL runtime 2023 R1 includes a new file handler named VisionJ to access Vision files without the need for native code. It also supports new library routines and improved existing routines and configurations.

<u>New file handler VisionJ</u>

A new value is supported for iscobol.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-performace 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 × 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
Total Time:	**36.81**	**22.64**	**21.85**	**15.84**

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 iscobol.visionj. For example, to create Vision files version four, set `iscobol.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 wkbuf-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.gui.implied_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.

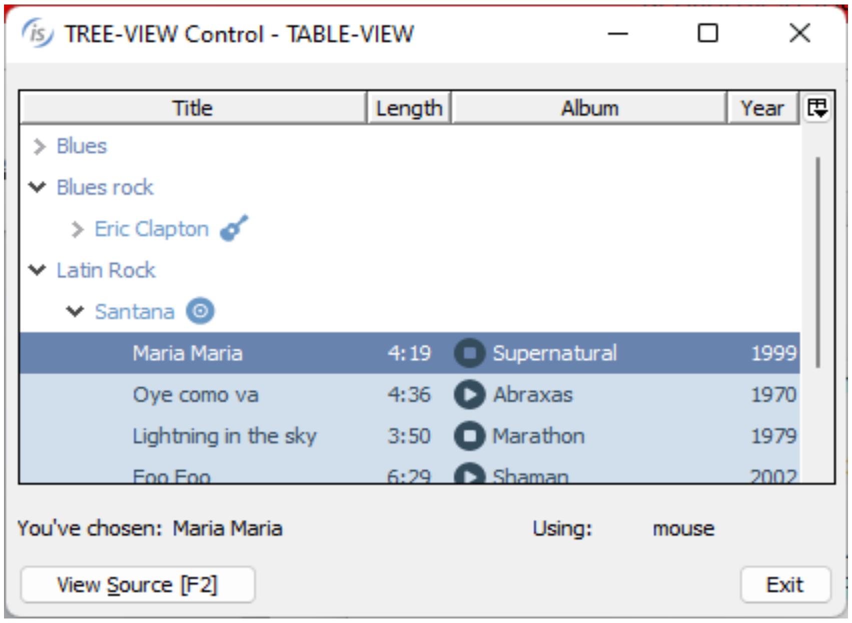
The following code snippet:

```
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.



Title	Length	Album	Year
Blues			
Blues rock			
Eric Clapton			
Latin Rock			
Santana			
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

You've chosen: Maria Maria   Using: mouse

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.

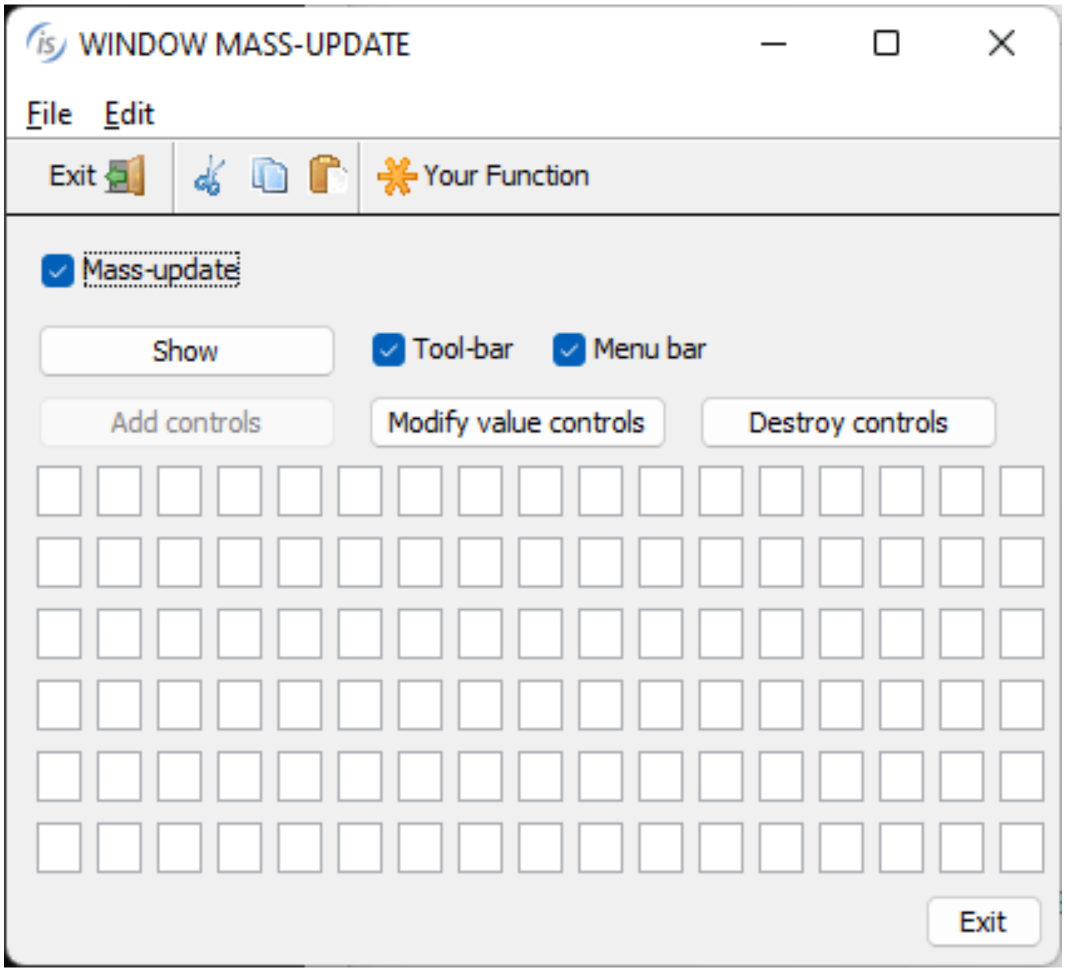
MASS-UPDATE on controls should still be used to ensure best performance.

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></soape
nv: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><StateProvinc
e>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><RegionN
ame
/><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.

<u>JUTIL</u>

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 iscobol.file.index.data_suffix and iscobol.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:USAddress" maxOccurs="2"/>
  </xsd:sequence>
  <xsd:attribute name="OrderDate" type="xsd:date"/>
 </xsd:complexType>
 <xsd:complexType name="USAddress">
  <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="hign"/>
      <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 'hign'.
         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).
```