



isCOBOL™ Evolve

isCOBOL Evolve 2021 Release 1 Overview

Copyright © 2021 Veryant LLC.

All rights reserved.

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

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

isCOBOL Evolve 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

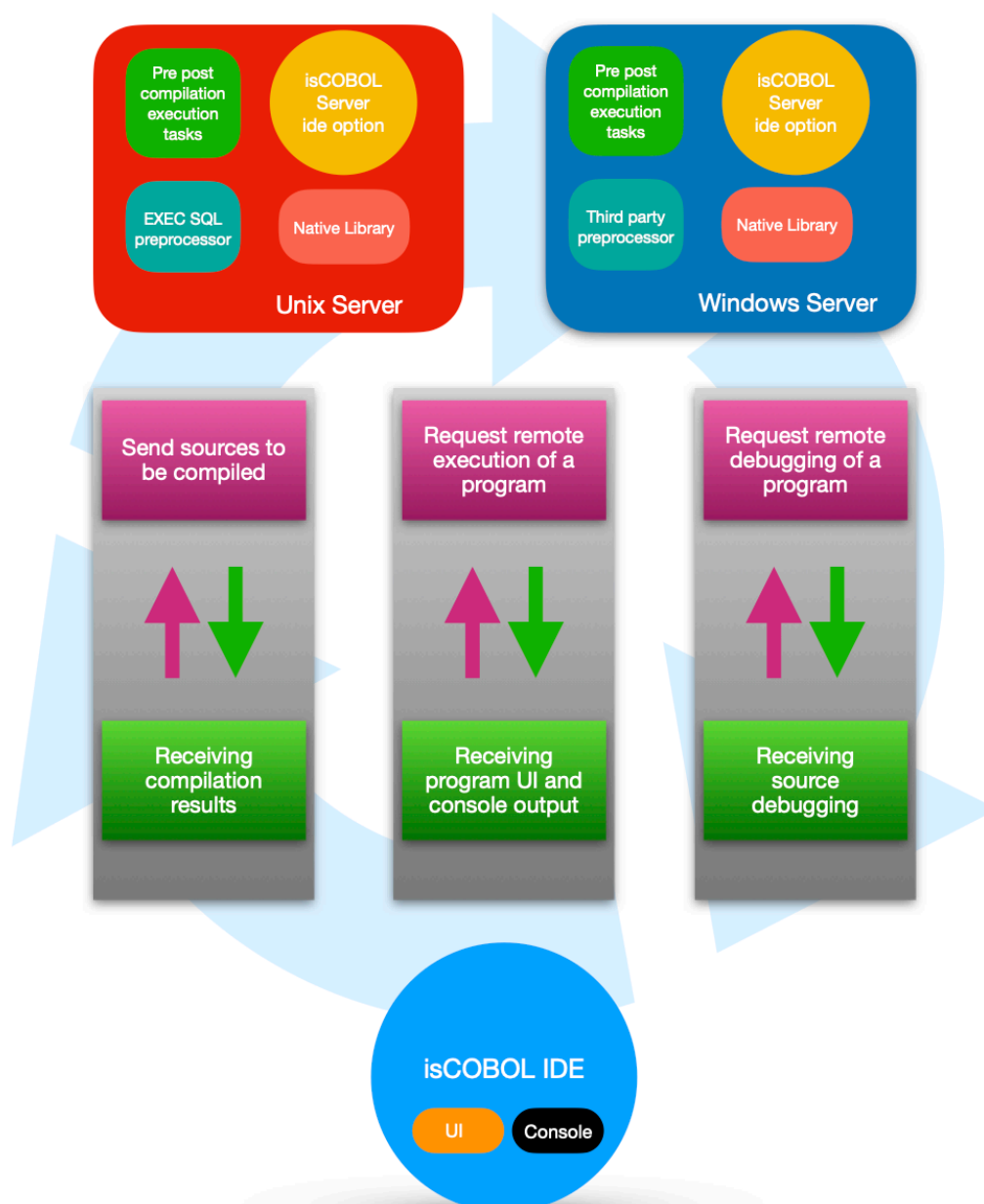
The isCOBOL IDE, now based on Eclipse 2020-06 (4.16), offers a new feature to easily work with remote projects. Remote projects are defined by having local source code, but needing remote compilation and execution. Typical scenarios for remote projects are batch programs that need to be executed on a 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.

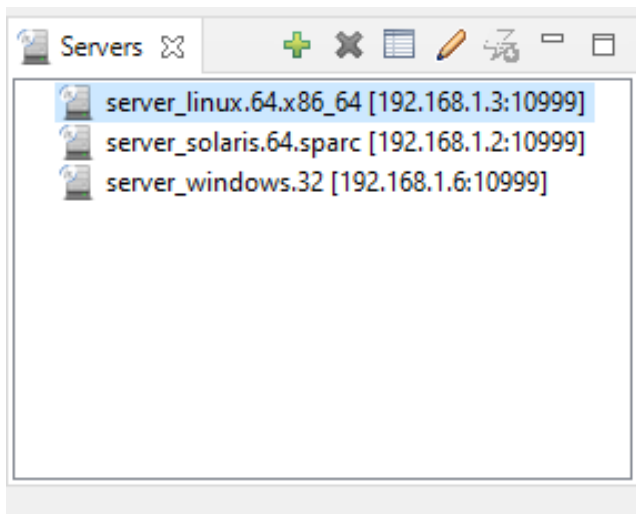
iscserver -ide

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 `isservice.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.

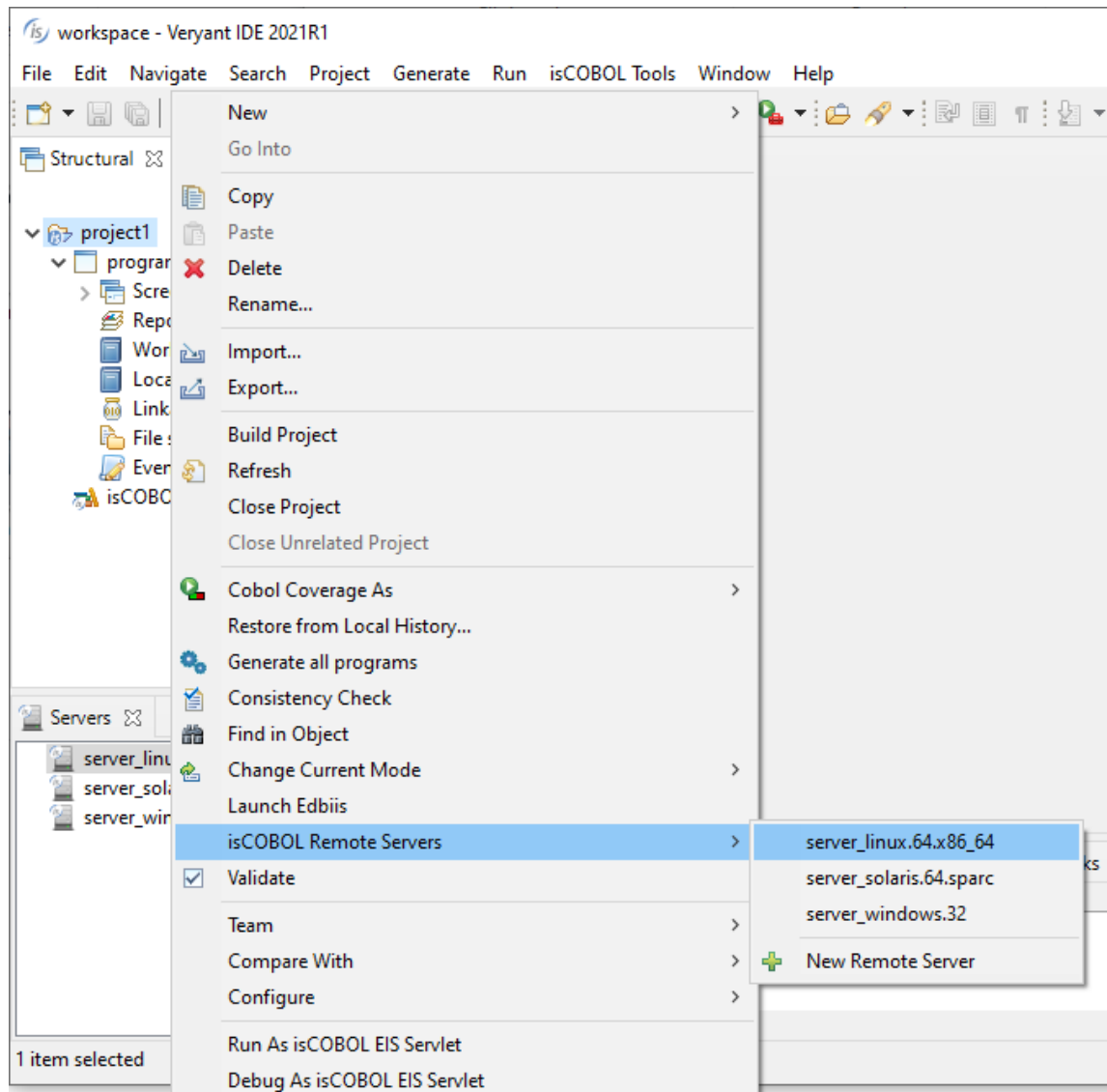
Figure 2. IDE's Servers view



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.

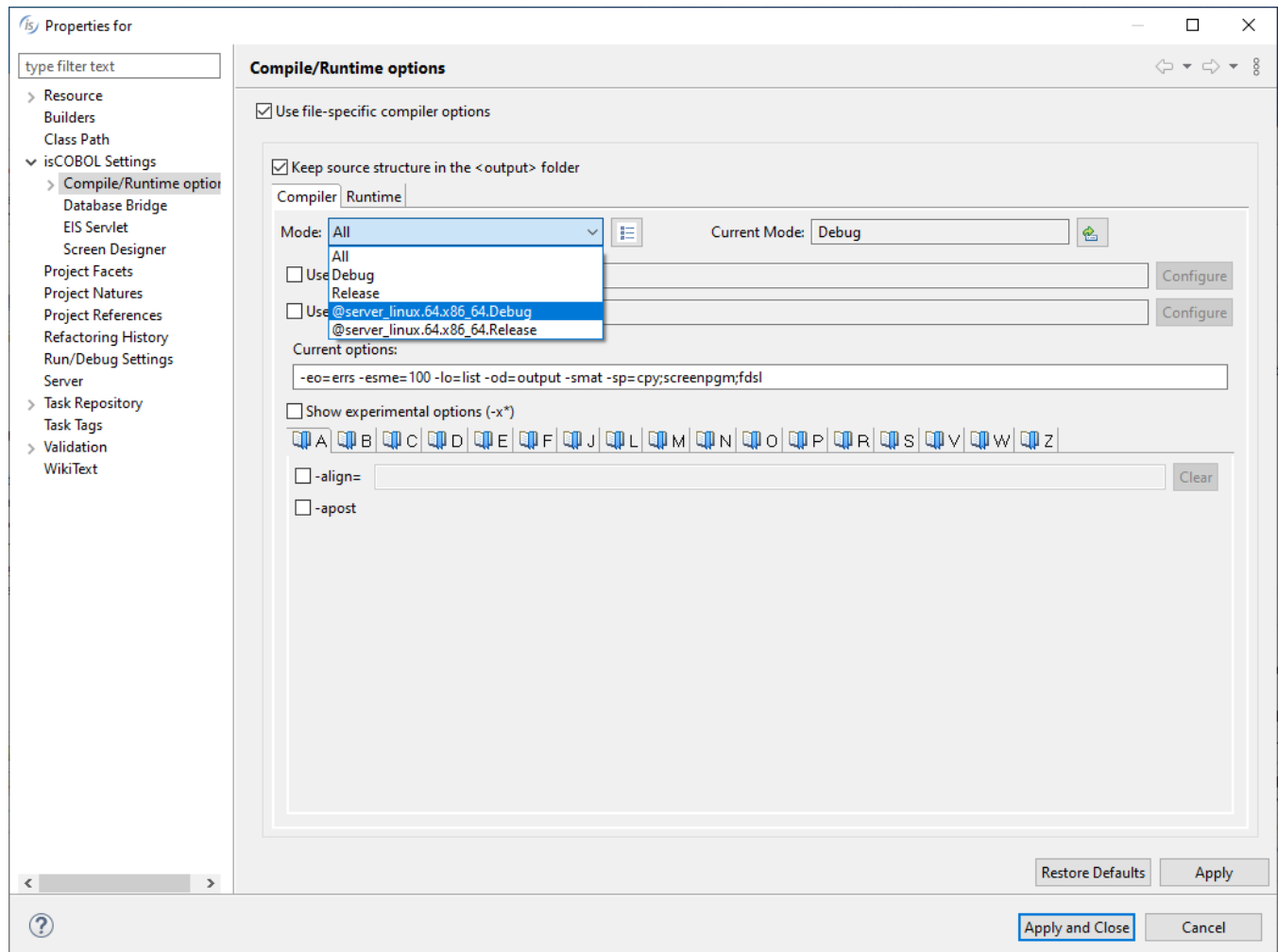
To bind a project to a remote server, right click on the project name in the isCOBOL Explorer and choose the desired server from the list in isCOBOL Remote Servers, as depicted in Figure 3, *Binding a project to a remote server*.

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.

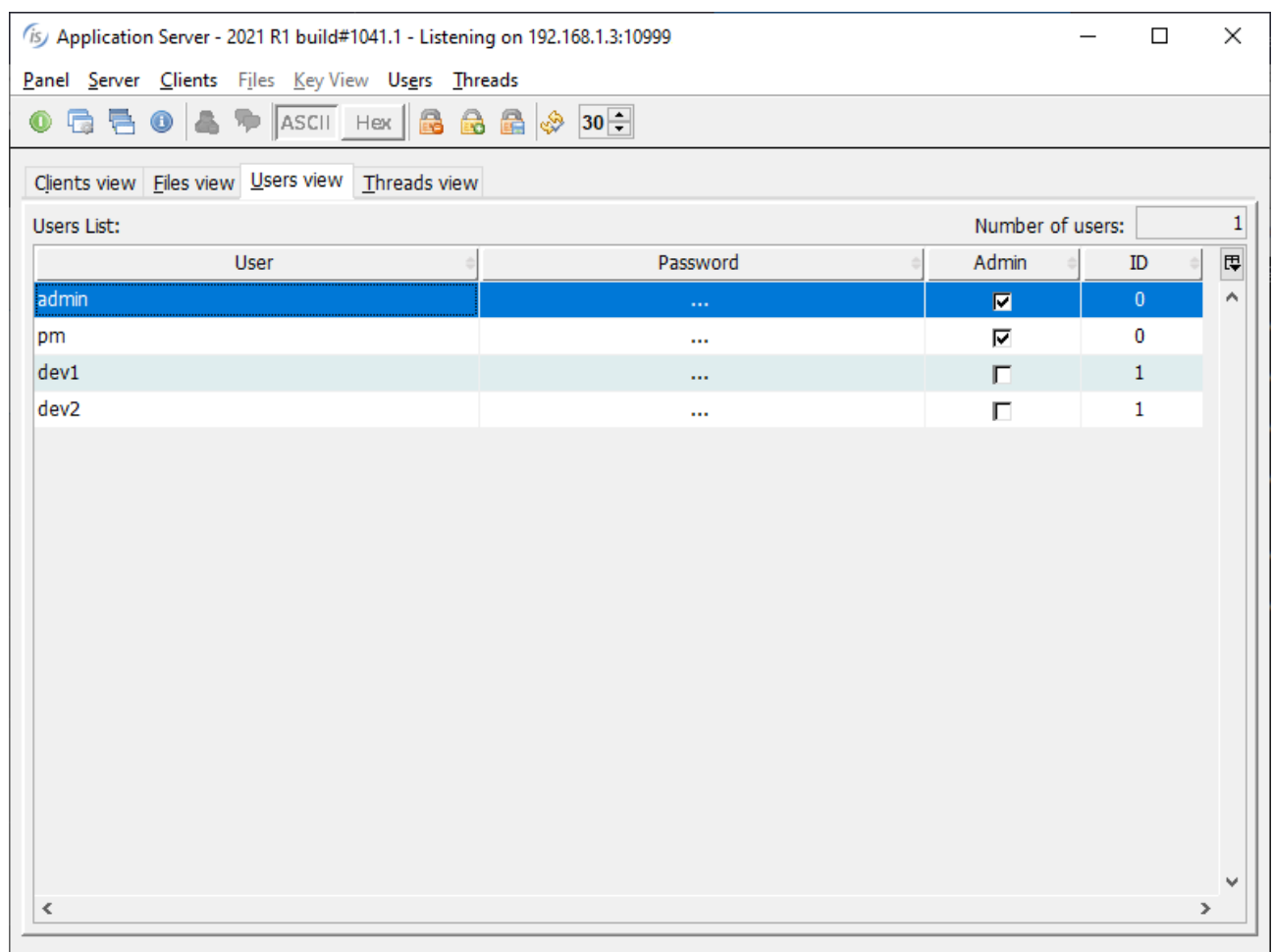
When you compile on a remote server, the IDE pushes the source files to the server. The source files are compiled on the server and the IDE receives the compilation results.

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



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.

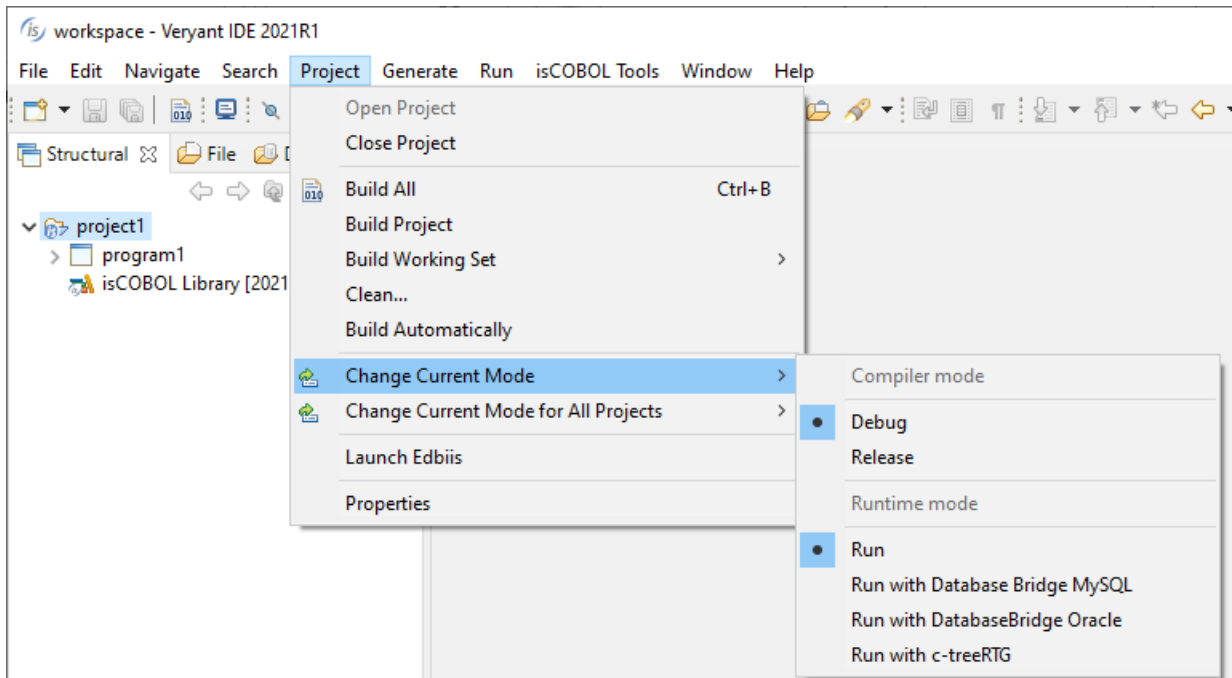
Figure 6, *Compile/Runtime options*, shows the separation of mode selection for the compile and runtime

Figure 6. Compile/Runtime options

The screenshot shows the 'Compile/Runtime options' dialog box. At the top, there is a checkbox labeled 'Keep source structure in the <output> folder' which is checked. Below this are two tabs: 'Compiler' and 'Runtime', with 'Compiler' currently selected. Under the 'Compiler' tab, there is a 'Mode:' dropdown menu set to 'All' and a 'Current Mode:' dropdown menu set to 'Debug'. Below these are two checkboxes: 'Use external preprocessor' and 'Use Remote Compiler', each followed by a text input field and a 'Configure' button. The 'Current options:' section contains a text input field with the value '-eo=errs -esme=100 -lo=list -od=output -smat -sp=cpy;screenpgm;fdsl'. Below this is a checkbox for 'Show experimental options (-x*)'. Underneath are 26 icons labeled A through Z, each with a small document icon. At the bottom, there are two checkboxes: '-align=' and '-apost', each followed by a text input field and a 'Clear' button.

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 SOURCE-COMPUTER 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, TIMESTAMPDIF, 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.html>

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

The screenshot displays the 'App Configuration' page in the WebClient application. The interface features a dark sidebar on the left with a search bar and navigation links: 'Server Config', 'isCOBOL De... /demo', and 'Create New App'. The main content area is titled 'App Configuration' and contains a form for configuring an application. The form includes the following fields:

- isCOBOL Server address**: 127.0.0.1
- isCOBOL Server port**: 10999
- Program name and arguments**: ISCONTROLSET
- Username**
- Password**
- LAF**: none
- Remote configuration**

Each field has a dropdown arrow icon, indicating that the values can be selected from a list or modified.

- 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.

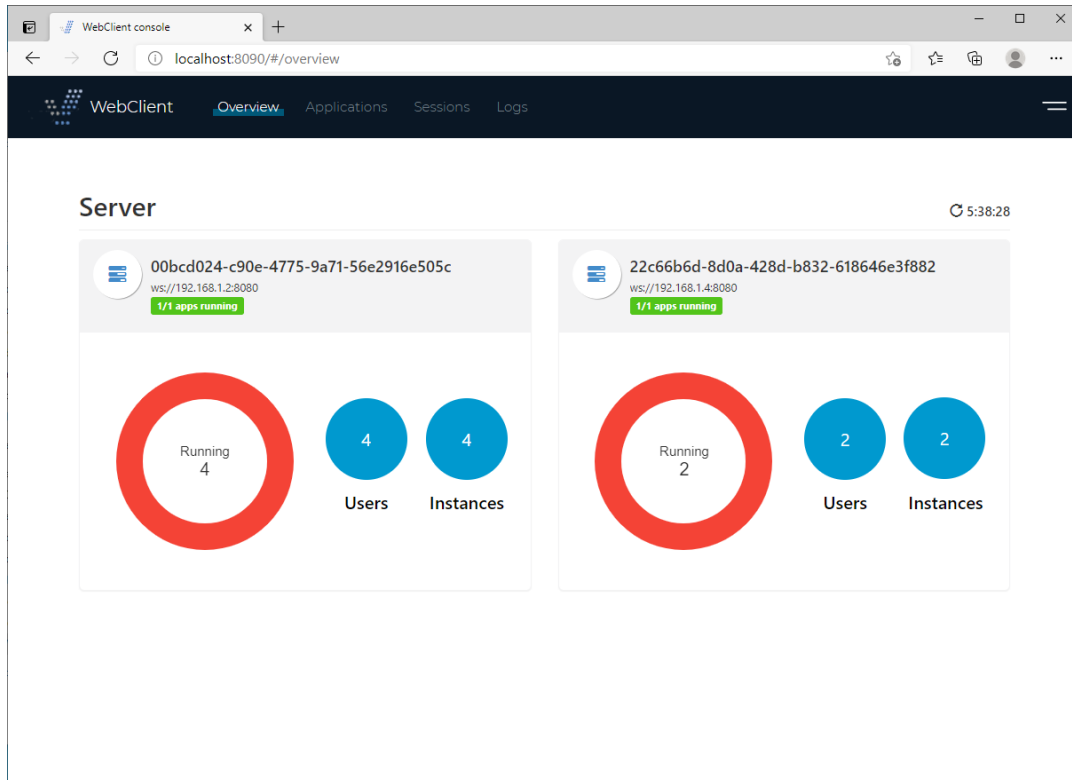
`webclient.server.websocketUrl = ws://localhost:8080,ws://server2:8080`

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

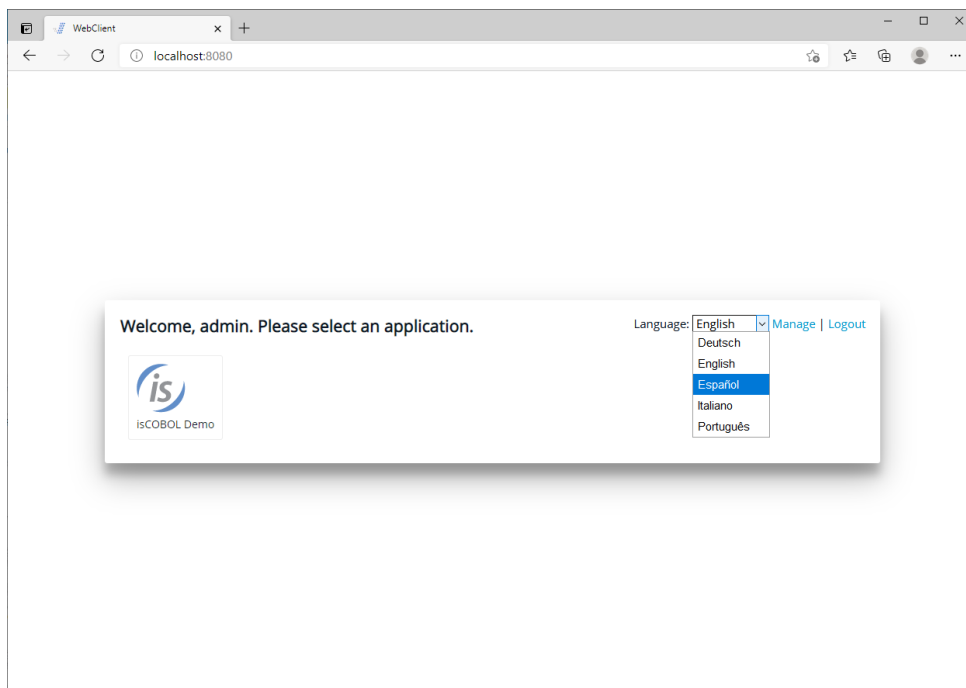
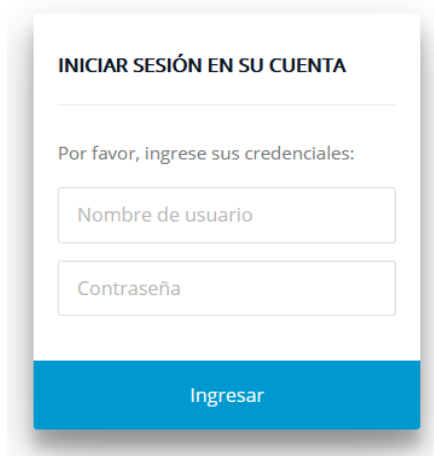


Figure 12 – *Spanish login*, shows login screen after Spanish has been selected.

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:

- Make a copy of `webclient-server.war` from the isCOBOL SDK to Tomcat's `webapps` folder
- In `webclient.properties` set the property `webclient.server.websocketUrl` to `ws://localhost:<port>` with the port that Tomcat is running on
- 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 tab-control 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-TAB-FOREGROUND-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.

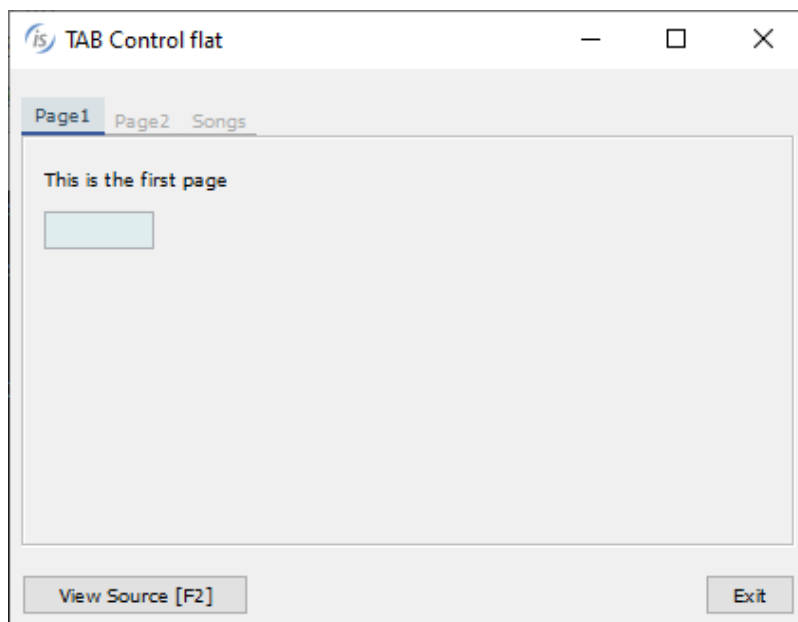
TAB-FLAT sets the flat style for pages.

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 **DISABLED-BACKGROUND-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

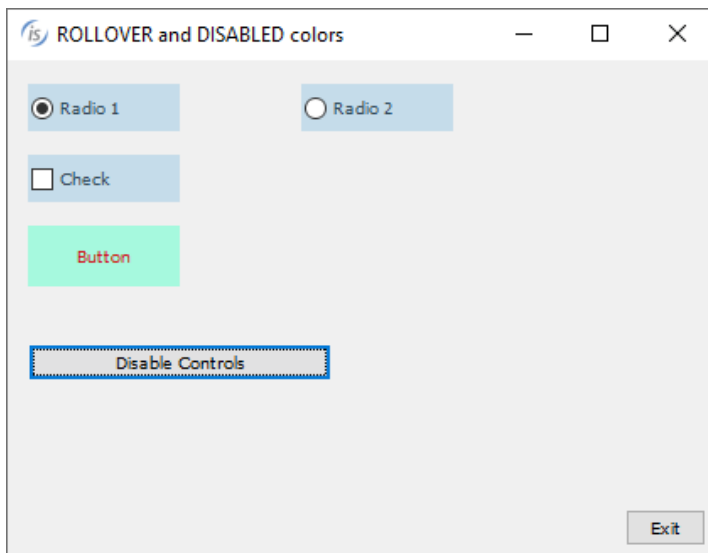
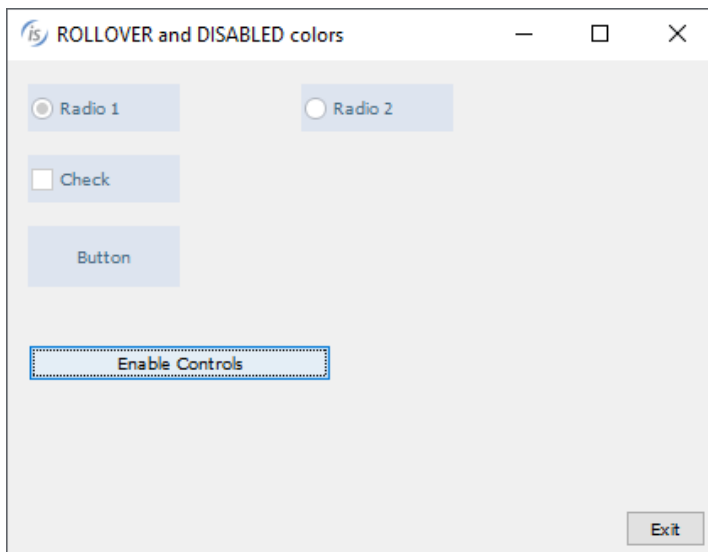


Figure 15. Disabled colors



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 **BITMAP-SCALE**:

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

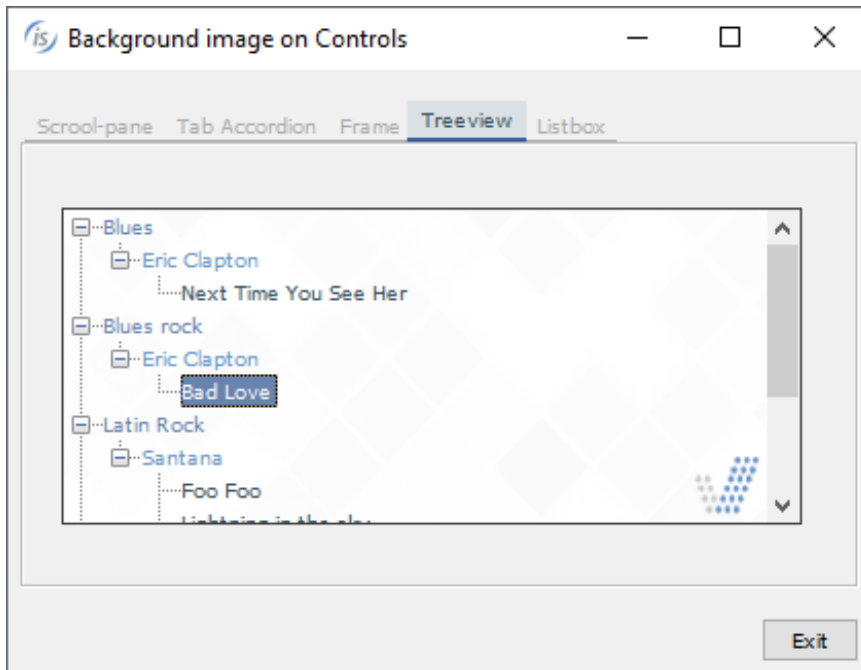
1 = to resize the image automatically to fit completely the control area. The aspect ratio may not be preserved with this option

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-snippet 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:

```

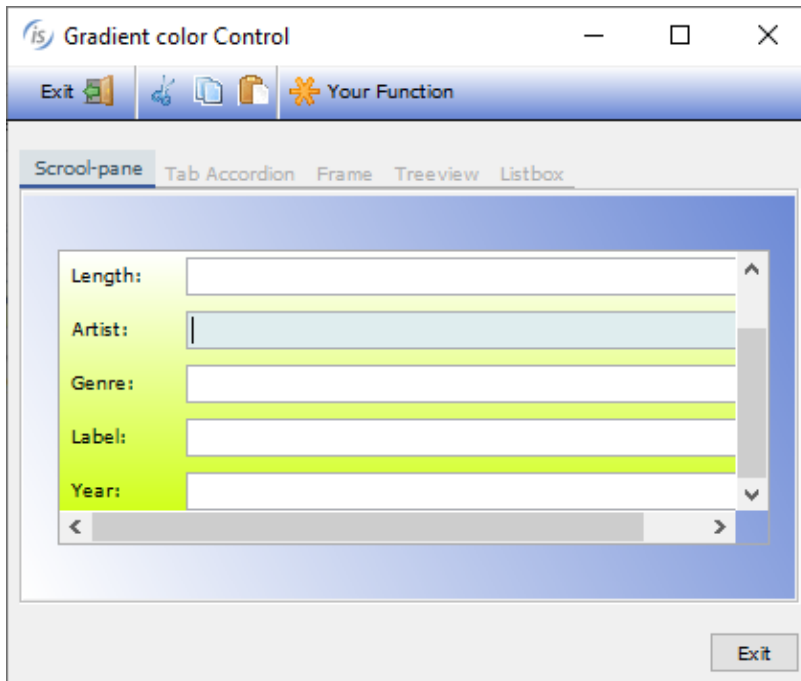
03 tb1-container tab-control
   line 2 col 2 lines 17 cells size 68 cells
   ...
   gradient-color-1 w-gradient-color-blue-1
   gradient-color-2 w-gradient-color-blue-2
   gradient-orientation gradient-northeast-to-southwest.
03 tb1-container-page1 tab-group Tb1-container tab-group-value 1.
05 scroll-pane-1 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:

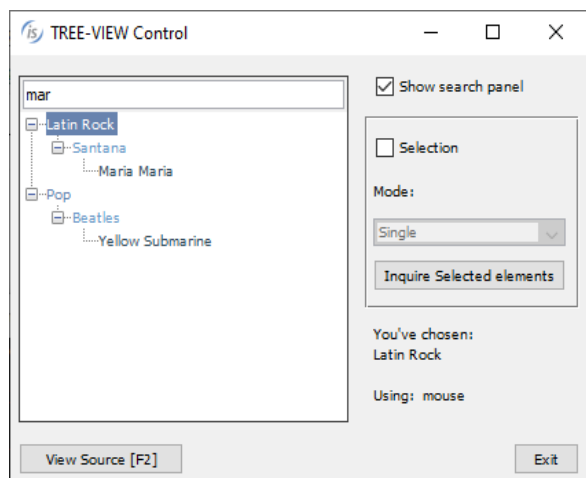
```
modify Tv1 search-panel 1
```

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



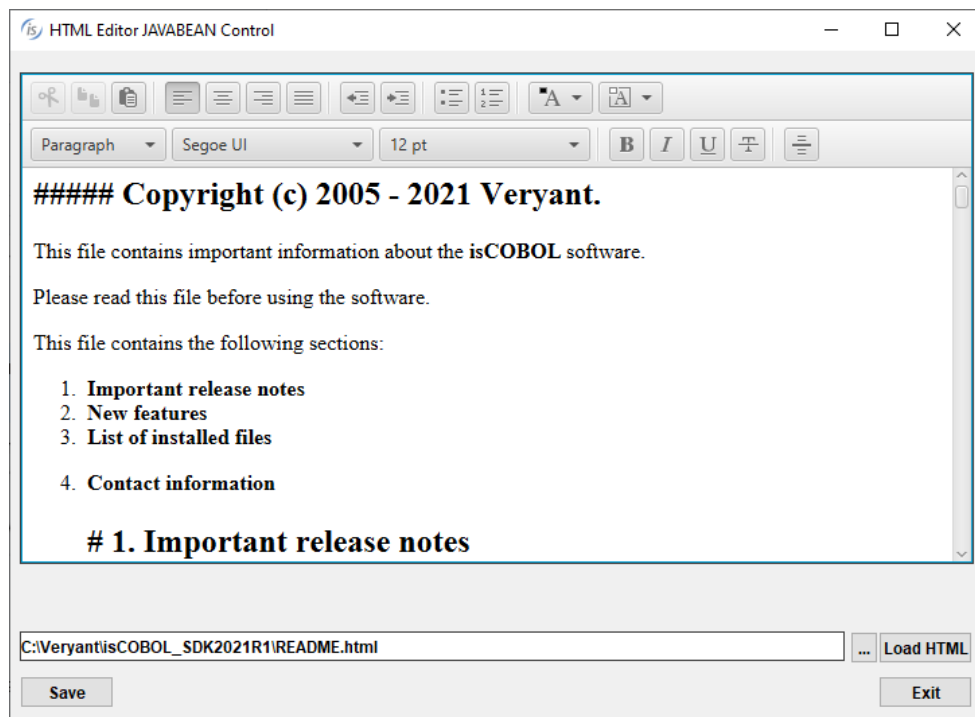
The java-bean control can now contain a JavaFX component, increasing GUI choices.

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\$OPENSABEBOX 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: W-CONTACTS" 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 data-items 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

iscobol.file.connector.program.rmc=path_where_rmc_is_located (set if rmc.exe is not located in PATH)

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... INPUT INDEX OUTPUT SEQ	19.11	8.97
statement SORT USING ... GIVING... INPUT 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-PBATCH56.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

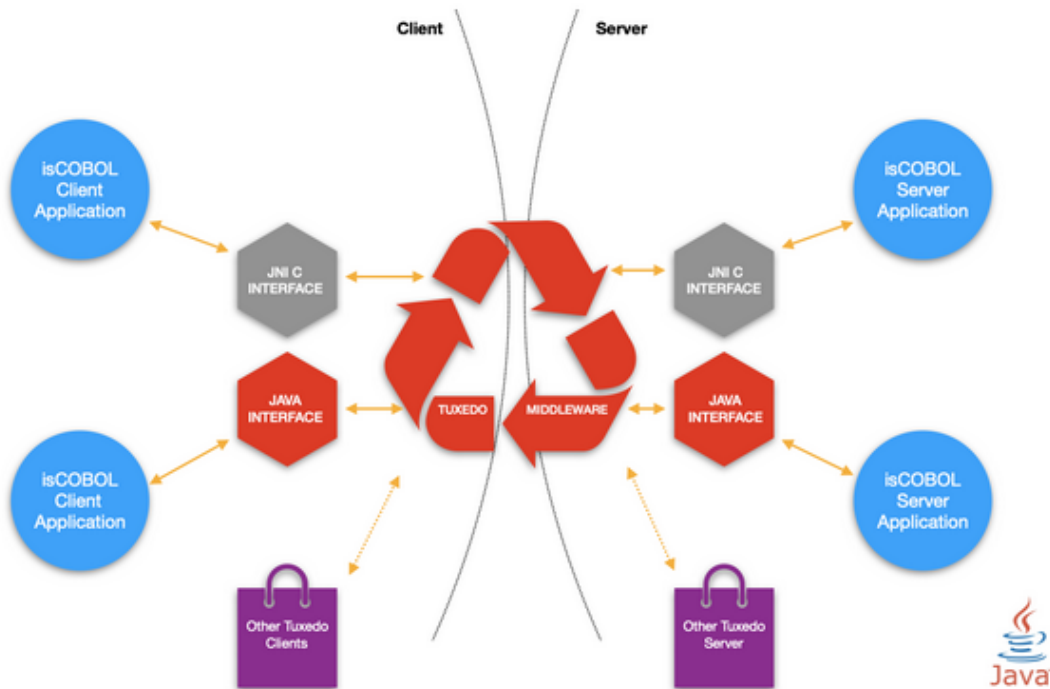
Tuxedo (Transactions for Unix, Extended for Distributed Operations) is a middleware platform used to manage distributed transaction processing in distributed computing environments. It distributes applications across multiple platforms, databases, and operating systems using message-based communications 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 user-customizable 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:

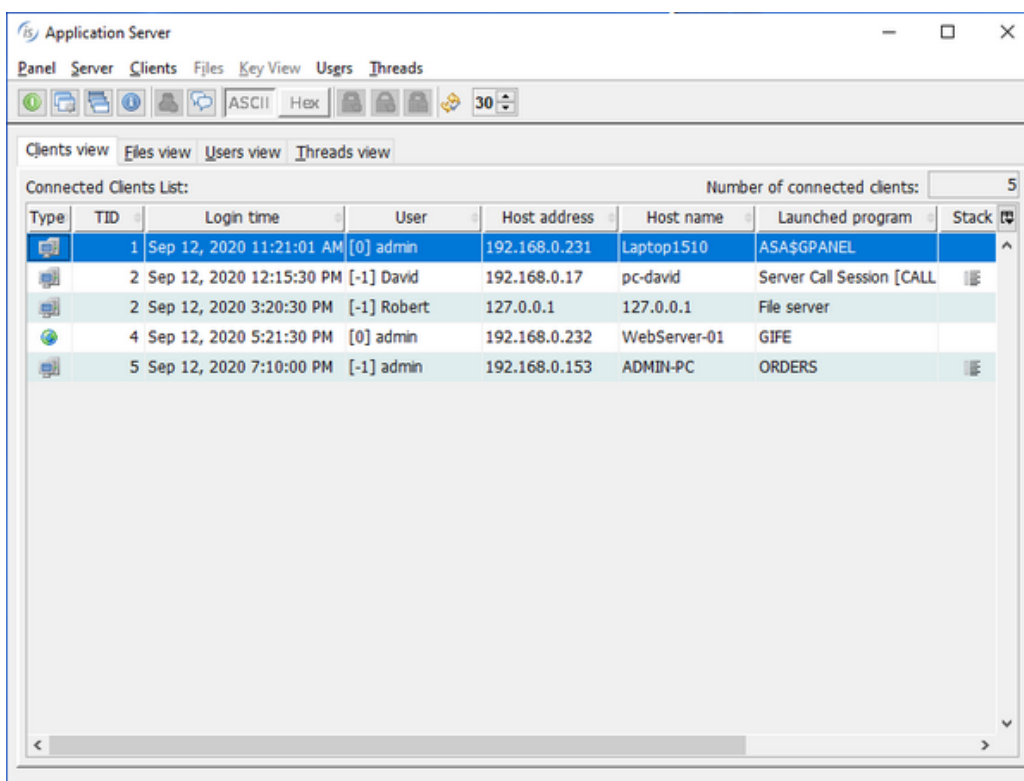
```
01 usr-logon-time pic x(16).
```

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

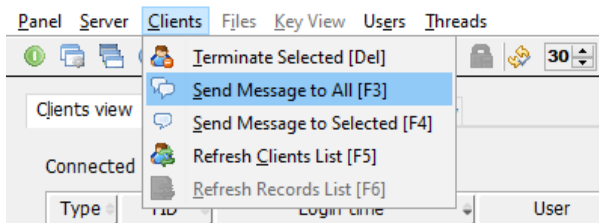


Figure 24. Send Message window

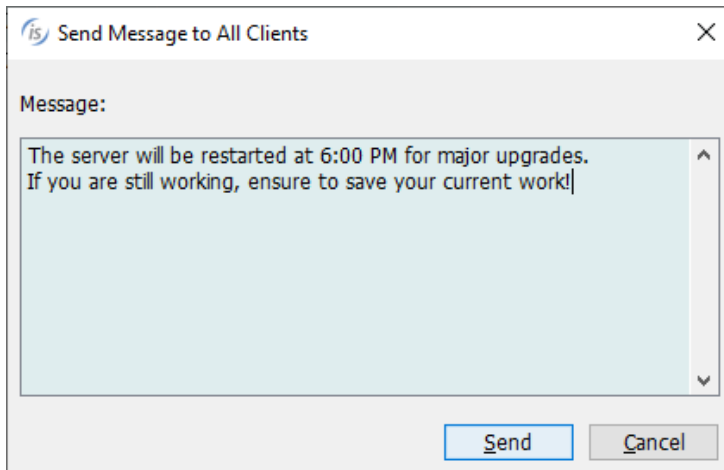
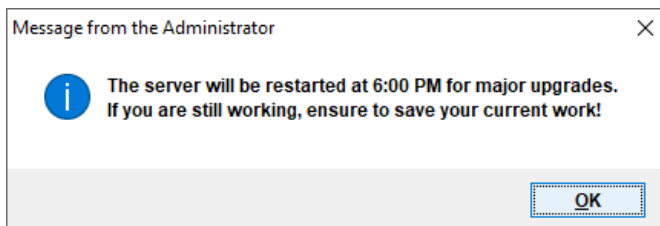


Figure 25. Message received from ThinClient



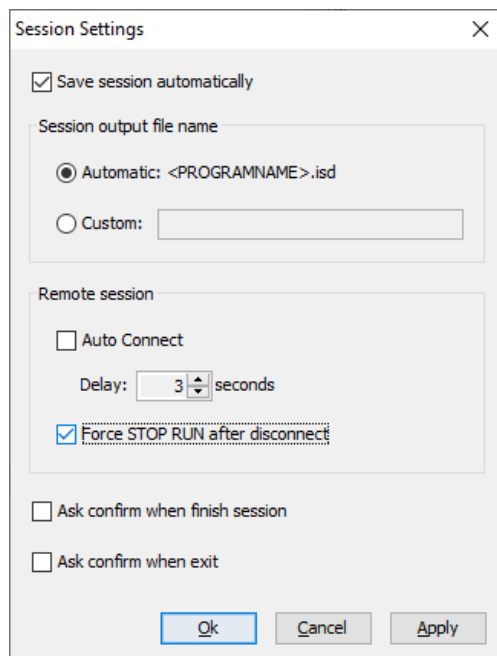
isCOBOL Debugger

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 `"isrun -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.

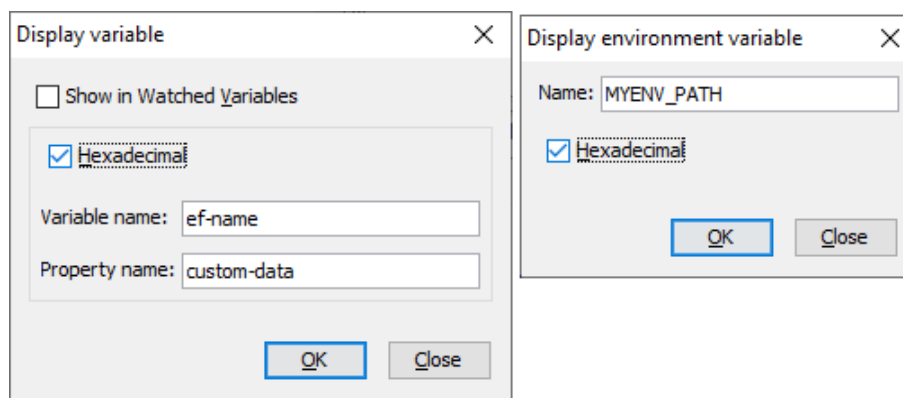
This is available in the debugger's command-line, using the commands:

```
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 (default true)

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

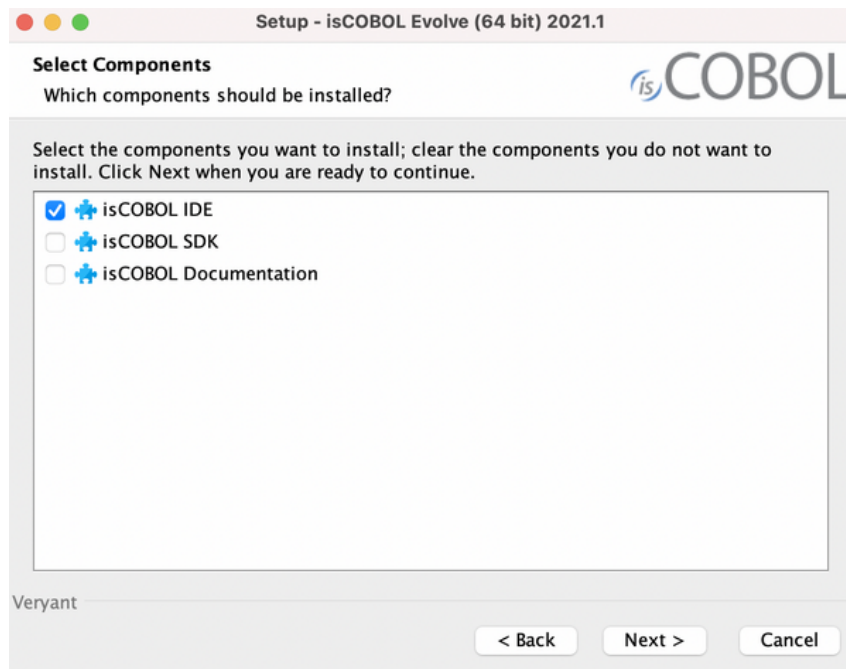
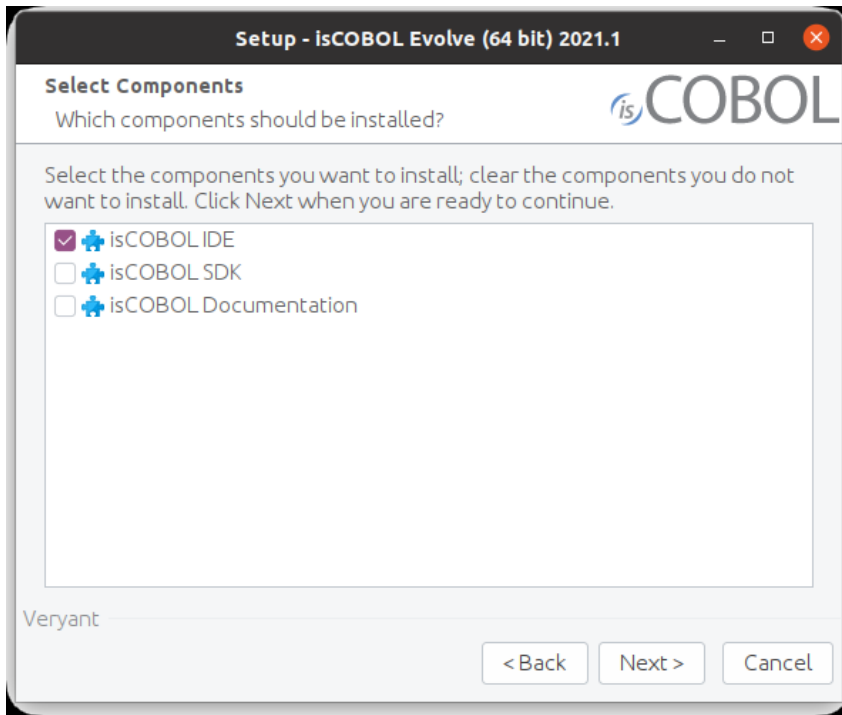


Figure 29. New Linux setup

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**:

Edit the `ctsrvr.cfg` configuration file and enable the plugin by removing the semicolon before **PLUGIN ctagent**

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 `cthttpd.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

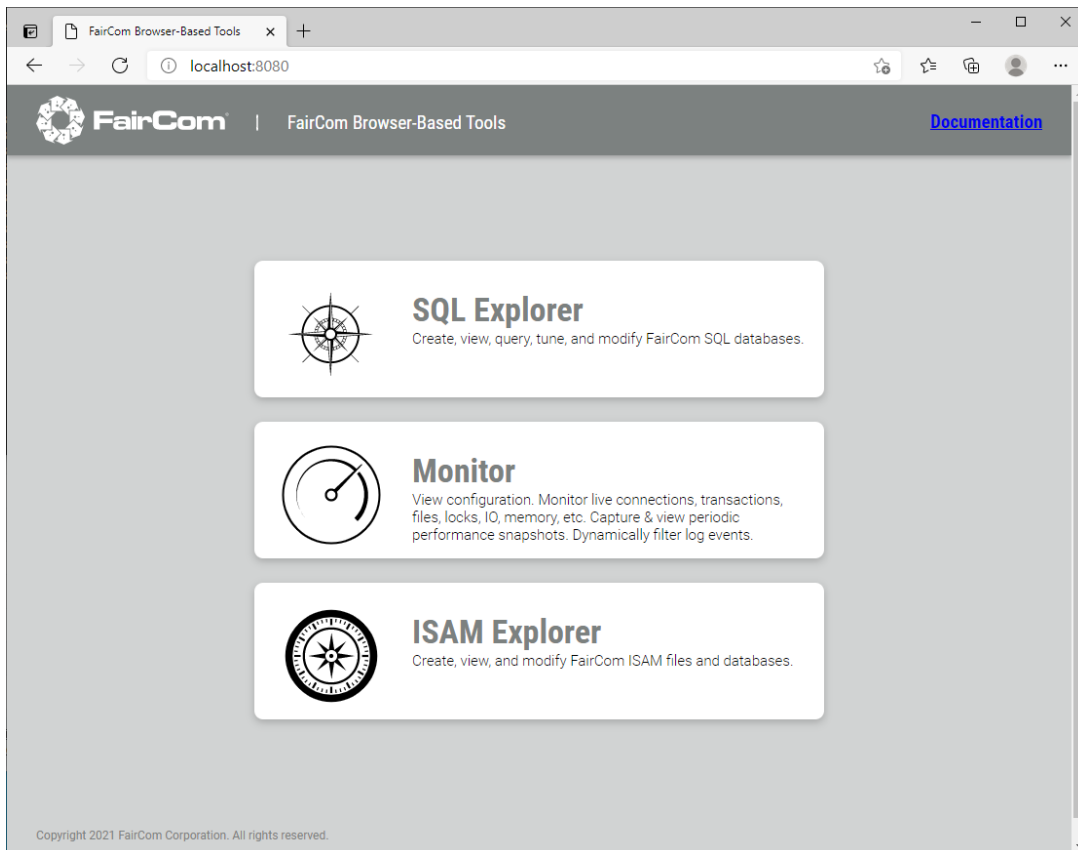


Figure 31, *Web SQL Explorer*, shows the layout of the browser-based SQL Explorer. This utility allows you to create, view, query, tune, and modify FairCom SQL databases.

Figure 31. Web SQL Explorer

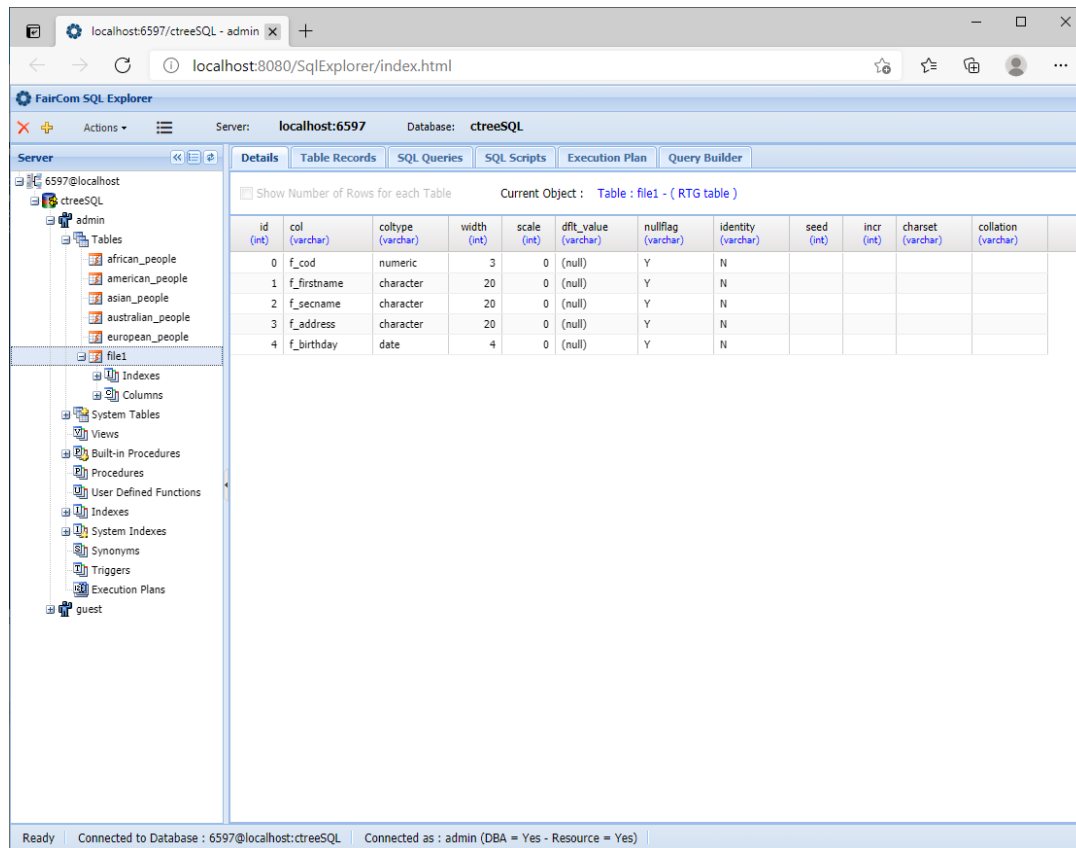


Figure 32, *Web Monitor*, shows the layout of the browser-based Monitor. This utility allows you to monitor live connections, transactions, files, locks, IO, memory, etc. It also allows you to view configurations, capture and view periodic performance snapshots and dynamically filter log events.

Figure 32. Web Monitor

The screenshot displays the FairCom Monitor web interface in a browser window. The address bar shows `localhost:8080/AceMonitor/index.html`. The interface includes a navigation bar with tabs for various monitoring functions. The 'Active Connections' tab is selected, showing a table of active connections. Below this, a 'User Snapshot for Task # : 22' is displayed, showing details for a specific task.

Active Connections Table:

#	Task...	User Name	Client IP Address	Node ID Info	Last Function	Act...	Last Request Ti...	Last TRANBEG T...	Logon Time	Fl...	Memory	Comm 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	FSHAREMM
2	24	ADMIN (Curre...	127.0.0.1	Web Server - ACEMonitor	USERINFO	yes	Apr 20, 2021 15...	n.a.	Apr 20, 2021 15...	0	59,168	FSHAREMM

User Snapshot for Task # : 22

#	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	fixlen	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 (M d,Y H:m:s)	Apr 20, 2021 15:37:17	
8	strntsum	Internal	user trntime sum^(d - H : m : s)	0 - 00 : 00 : 00	
9	strntmax	Internal	user trntime max^(d - H : m : s)	0 - 00 : 00 : 00	

Ready | 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 the Web ISAM Explorer interface. The browser address bar indicates the URL is localhost:8080/IsamExplorer/index.html. The interface has a sidebar on the left showing a tree view of the database structure, including 'FAIRCOMS@localhost', 'ctreeSQL', and 'file1'. The main area displays a table of records for 'file1' in the 'ctreeSQL' database. The table has five columns: F_COD (CT_NUMBER), F_FIRSTNAME (CT_FSTRING), F_SECNAME (CT_FSTRING), F_ADDRESS (CT_FSTRING), and F_BIRTHDAY (CT_DATE). The table shows 26 records, with the first two being 'Smith' and 'Johnson', and the rest being 'AA'. The status bar at the bottom indicates 'Ready' and 'Connected to Server: FAIRCOMS@localhost'.

F_COD (CT_NUMBER)	F_FIRSTNAME (CT_FSTRING)	F_SECNAME (CT_FSTRING)	F_ADDRESS (CT_FSTRING)	F_BIRTHDAY (CT_DATE)
1	Smith	Albert	49 West 44th St.	10/11/1971
2	Johnson	Alan	2248 Palm Spring Rd.	01/14/1968
3	AA	BB	CC	01/03/1954
4	AA	BB	CC	01/03/1954
5	AA	BB	CC	01/03/1954
6	AA	BB	CC	01/03/1954
7	AA	BB	CC	01/03/1954
8	AA	BB	CC	01/03/1954
9	AA	BB	CC	01/03/1954
10	AA	BB	CC	01/03/1954
11	AA	BB	CC	01/03/1954
12	AA	BB	CC	01/03/1954
13	AA	BB	CC	01/03/1954
14	AA	BB	CC	01/03/1954
15	AA	BB	CC	01/03/1954
16	AA	BB	CC	01/03/1954
17	AA	BB	CC	01/03/1954
18	AA	BB	CC	01/03/1954
19	AA	BB	CC	01/03/1954
20	AA	BB	CC	01/03/1954
21	AA	BB	CC	01/03/1954
22	AA	BB	CC	01/03/1954
23	AA	BB	CC	01/03/1954
24	AA	BB	CC	01/03/1954
25	AA	BB	CC	01/03/1954
26	AA	BB	CC	01/03/1954

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

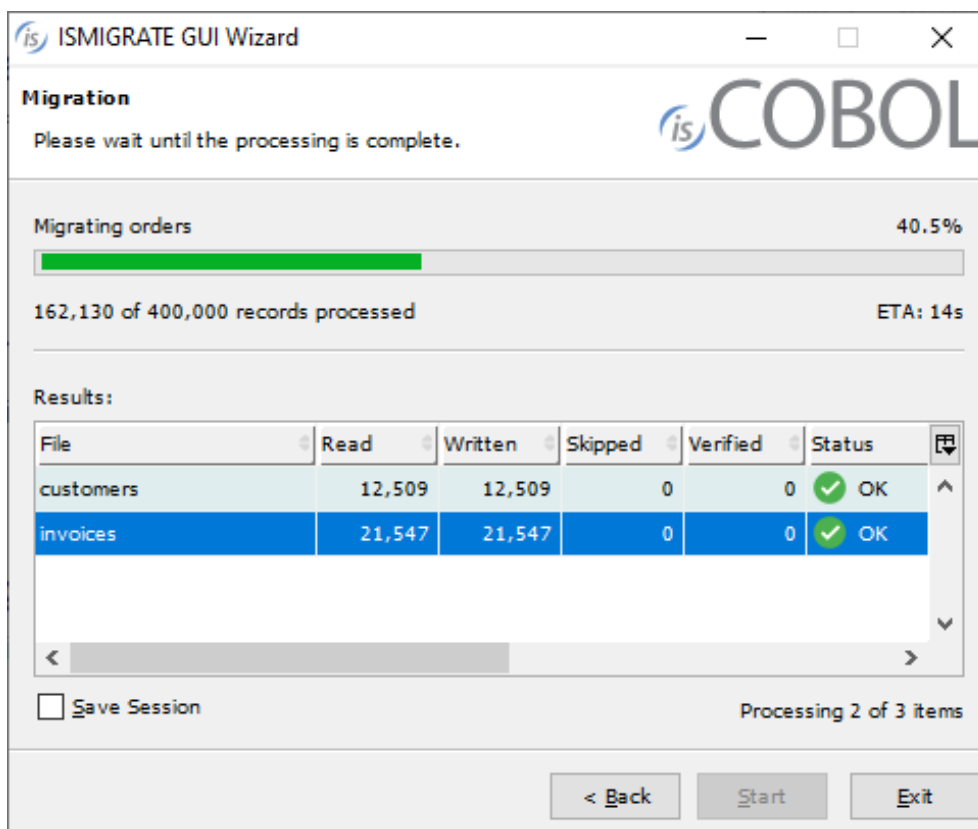
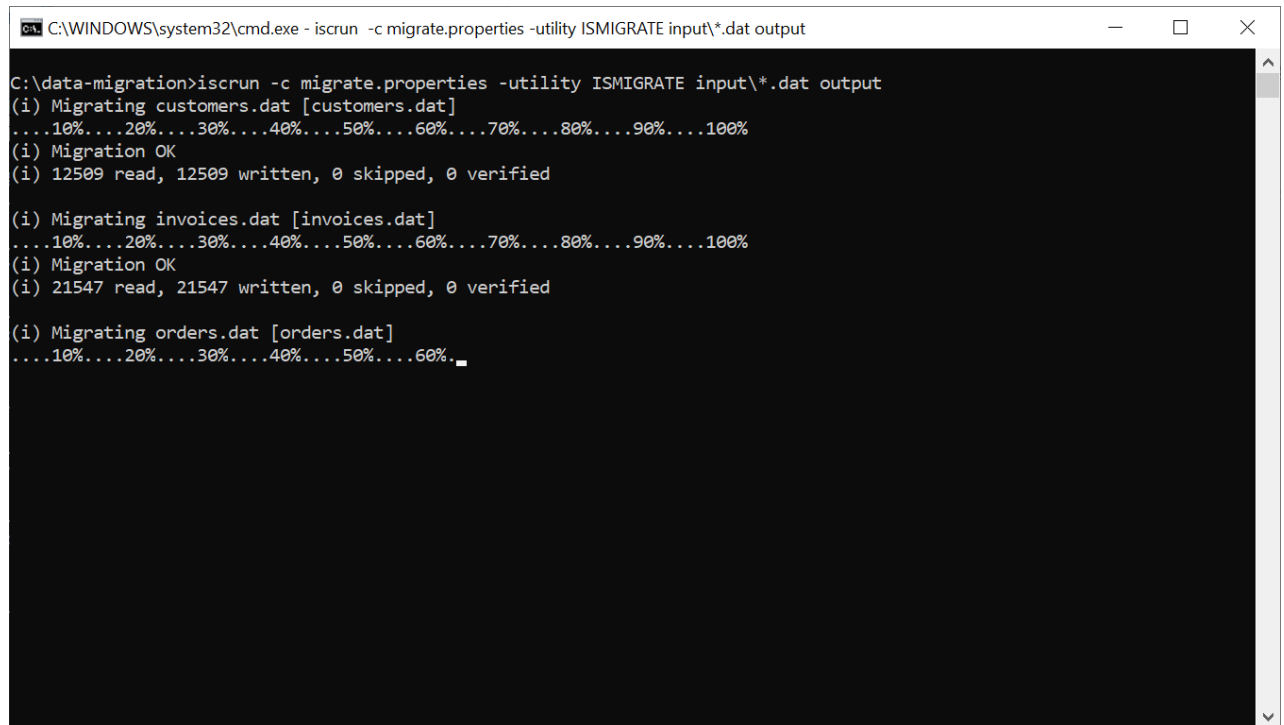


Figure 35, *ISMIGRATE command-line progress*, shows the same migration running in command-line mode

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 isclclient 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

