isCOBOL™ Evolve

isCOBOL Evolve 2026 Release 1 Overview

isCOBOL Evolve 2026 Release 1 Overview

Introduction

Veryant is pleased to announce the latest release of isCOBOL Evolve, isCOBOL Evolve 2026 R1.

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

isCOBOL Evolve 2026 R1 introduces a new product named NoSQL Bridge that allows access to indexed data files using HTTP requests in JSON format.

The 2026R1 release improves GUI components by supporting the Google's Material Design style on entry-field controls and offers other graphical improvements.

A new Apache Maven plugin is available to provide complete support for compiling isCOBOL applications using the Maven build system.

isCOBOL Debugger can now set breakpoints while the program is running.

The updated C-Tree RTG v5 adds the ability to increase the record length of an indexed file and add new keys without needing to migrate existing records.

Details on these enhancements and updates are included below.

NoSQL Bridge

NoSQL Bridge is a set of stateless REST API functions that provide access to indexed data files, supported by isCOBOL Evolve, through HTTP requests using JSON as a data format. This can ease the integration of third-party software with your database, without requiring you to write backend code.

The NoSQL Bridge product is available for installation on Windows with a dedicated MSI setup, and on Linux using the .tar.gz format. A separate license is required to use the product, and the license key must be configured in the iscobol.properties configuration file, as follows:

```
iscobol.nosql.license.2026=xxx...
```

NoSQL Bridge embeds the isCOBOL runtime to access data file. It supports all indexed file formats and uses the EFD dictionaries (the XML files generated by the Compiler when using the -efd option) that describe the file structure.

NoSQL Bridge comes with a lightweight Jetty-based web server that can be started in the foreground mode with the nosqlc command or installed as a service with the nosql command. The following commands show the different ways it can be started:

```
nosqlc
nosql –install
nosql -start
```

Alternatively, you can install the veryant-nsb.war library in any servlet container like Tomcat.

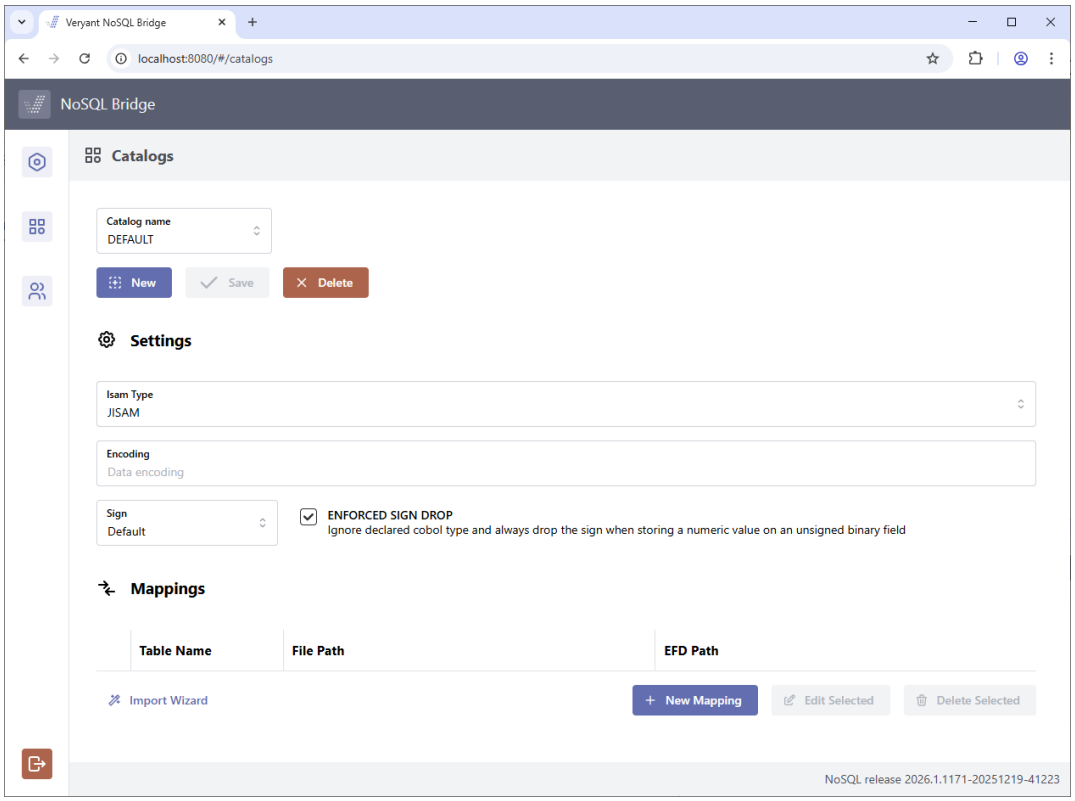
Admin configuration

Once the web server is started and active, you can browse to http://server-IP:port to access the admin pages that let you configure users and datasets, as well as change the admin password.

After logging in as ‘admin’ with password ‘admin’, you land on the “Catalogs” configuration.

This user interface is depicted in Figure 1, *NoSQL Catalogs configuration*.

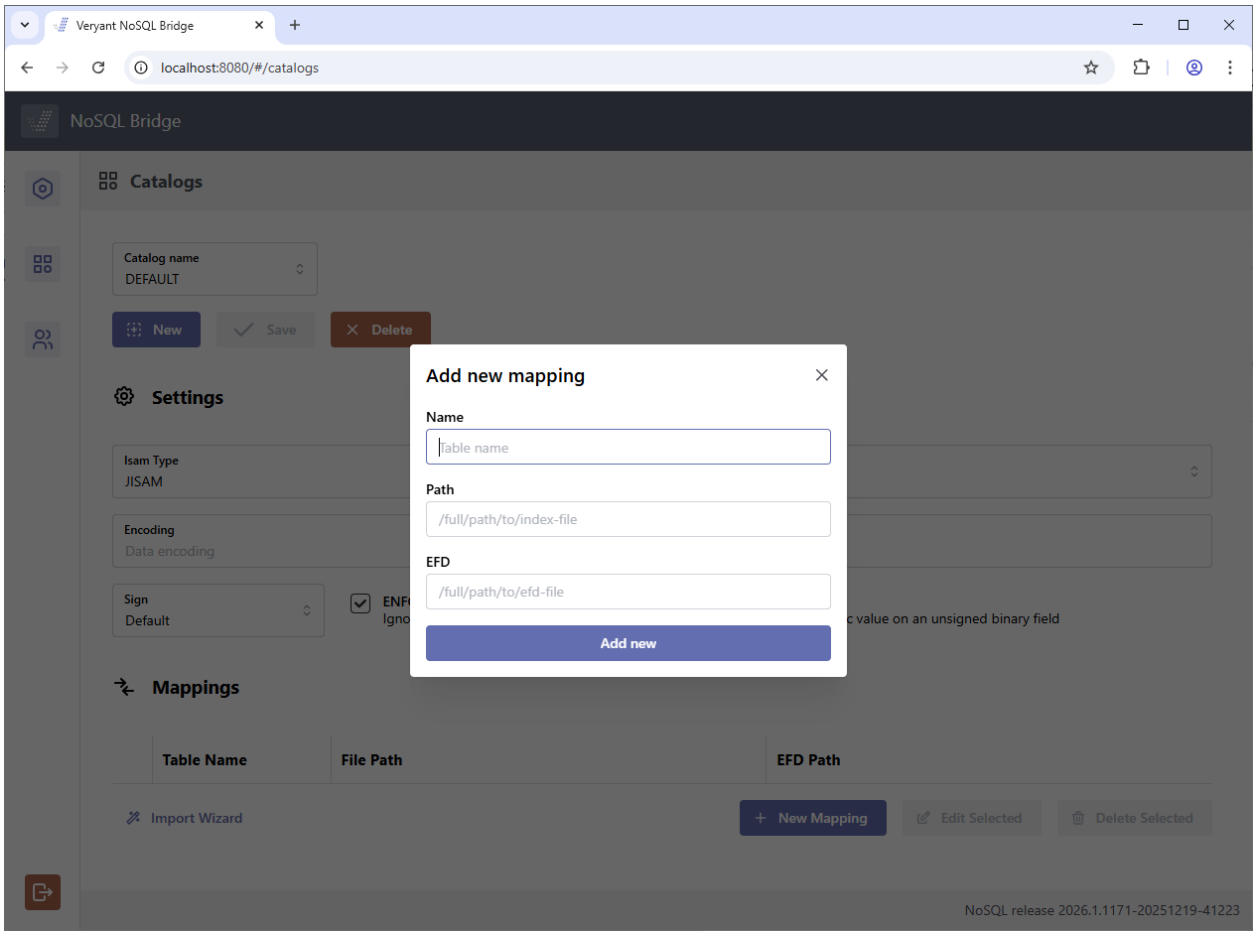
Figure 1. NoSQL Catalogs configuration.



A default catalog named DEFAULT is automatically created, but you can add additional catalogs by clicking the New button. In each catalog you can specify the indexed file handler (JISAM, VisionJ, etc, …), the character encoding and the sign convention. Then you must provide mappings between physical files and their dictionaries. For every mapping, you must specify: the logical name clients will use to identify the file, the file's physical location, and the location of the EFD dictionary.

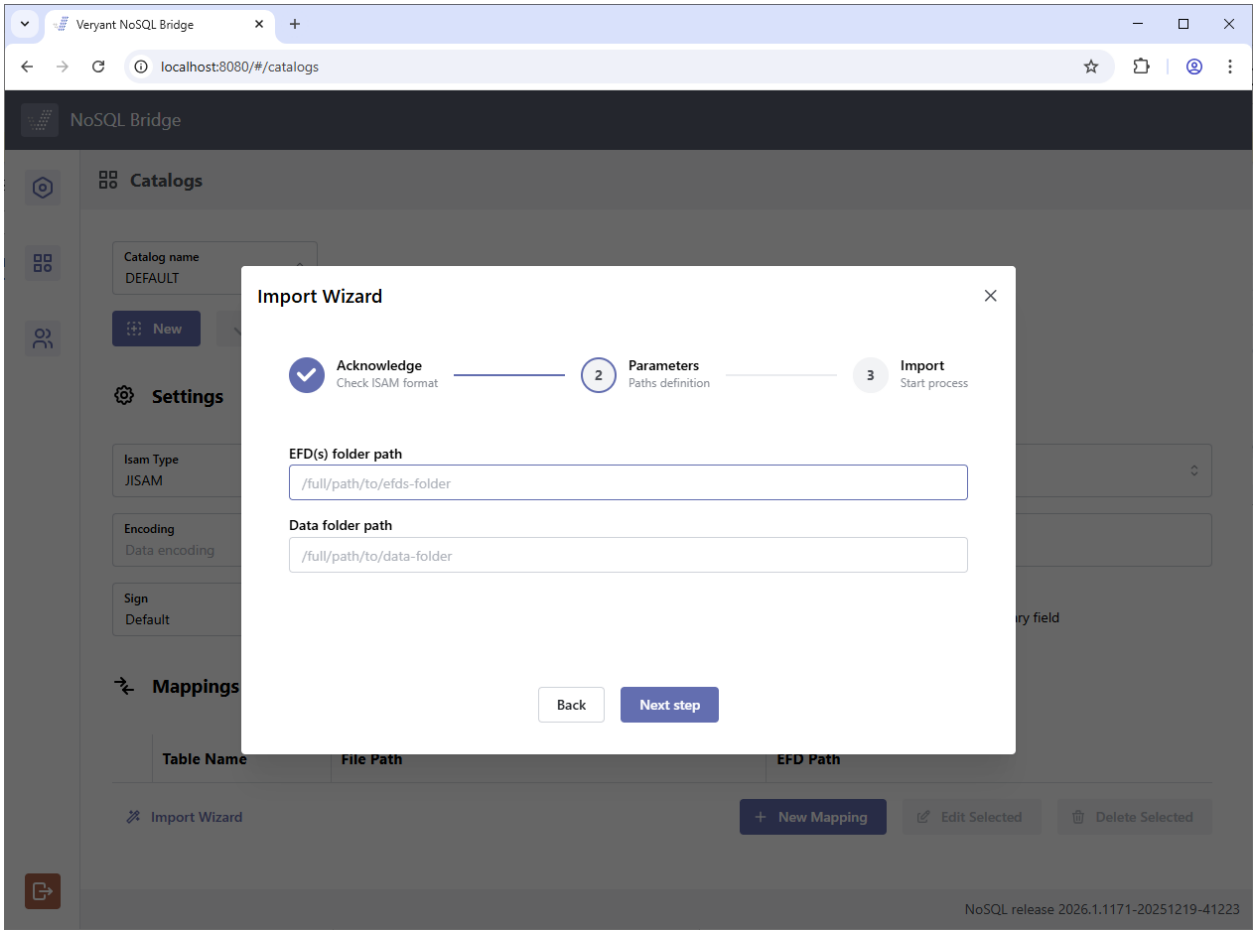
Mapping is depicted in Figure 2, *NoSQL New mapping*.

Figure 2. NoSQL New mapping.



The import wizard streamlines this process by allowing you to select the directories where data files and dictionaries are located. NoSQL Bridge automatically generates the mappings according to the contents of these directories, as shown in Figure 3, *NoSQL Mappings import wizard*.

Figure 3. NoSQL Mappings import wizard.

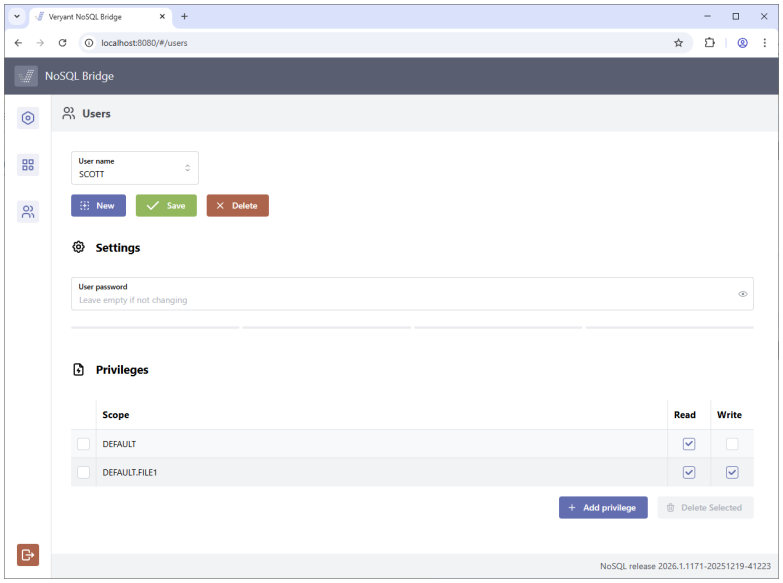


The administration portal offers two other pages that you can reach using the buttons on the left: Admin password change (depicted in Figure 4, *NoSQL Admin password*) and Users configuration. In the "Users" page you can define additional users, set the password, and set permissions on catalogs and mappings. Figure 5, *NoSQL New user*, depicts the creation of a new user and Figure 6, *NoSQL User permissions*, shows how the new user is given read-only access to each file in the DEFAULT catalog, except FILE1 where write permissions are also granted.

Figure 4. NoSQL Admin password.

Figure 5. NoSQL New user.

Figure 6. NoSQL User permissions.



HTTP requests

HTTP requests can be performed with any HTTP client, such as Postman.

NoSQL Bridge accepts HTTP POST requests with a JSON body for parameters and returns JSON objects. Security is handled by Basic Authentication, where credentials are passed in the HTTP request headers. Refer to the documentation for the details on JSON format for requests and responses.

Here are a few examples of the protocol:

To get the description (file structure) of FILE1 mapped in the DEFAULT catalog, send the following request:

Url:	http://server:port/DescribeTable

Body:	<pre>{
	  "tableName": "FILE1"
	}</pre>

The following is the response:

```
{
    "tableName": "FILE1",
    "readable": true,
    "writable": true,
    "tableDefinition": {
        "keys": {
            "F_REC_key0": {
                "paths": [
                    "F_REC.F_COD"
                ],
                "primary": true,
                "duplicates": false
            }
        },
        "recordDefinitions": {
            "F_REC": {
                "type": "object",
                "fieldDefinitions": {
                    "F_ADDRESS": {
                        "type": "string",
                        "length": 20
                    },
                    "F_BIRTHDAY": {
                        "type": "number",
                        "signed": false,
                        "truncation": true,
```

```
                    "signDrop": true,
                    "digits": 8
                },
                "F_COD": {
                    "type": "number",
                    "signed": false,
                    "truncation": true,
                    "signDrop": true,
                    "digits": 3
                },
                "F_FIRSTNAME": {
                    "type": "string",
                    "length": 20
                },
                "F_SECNAME": {
                    "type": "string",
                    "length": 20
                }
            }
        }
    }
}
```

To write a record to FILE1 on the TEST catalog, send the following request:

Url:	http://server:port/PutItem
Body:	`{ "tableName": "DEFAULT.FILE1", "item": { "F_REC": { "F_COD": 1, "F_FIRSTNAME": "John", "F_SECNAME": "Doe", "F_ADDRESS": "20 Cooper Square, New York, NY 10003, USA" } } }`

If the write operation is successful, the NoSQL Bridge returns a 200 HTTP response code, with no body.

The following are the functions currently supported by NoSQL Bridge:

DescribeTable: returns the file structure, with the list of fields, their data types and the structure of keys.

GetItem: returns the content of a single record (simulates READ KEY operation)

PutItem: adds a new record or updates it if the key already exists (simulates WRITE INVALID REWRITE)

UpdateItem: updates a record through expressions like SET field=value. Unlike the PutItem operation that required all fields to be specified, UpdateItems can accept a subset of fields

DeleteItem: deletes a record

Query: returns a list of records given search criteria (i.e. all records where the ID field is > 100)

GUI enhancements

isCOBOL Evolve 2026 R1 supports Google's Material Design styles on entry-field controls, provides better integration with LAF used when running, and offers other graphical improvements.

Entry-fields with Material Design

The new property MATERIAL-DESIGN allows you to give Entry-fields the Material Design style that is commonly used on web and mobile applications. It is not only a matter of appearance, but there are also specific runtime behaviors when the user interacts with these fields:

- When the field gets the focus, the border color changes, and the placeholder text becomes a floating label that moves above the field to make room for the content.
- When the field loses the focus, the floating label remains above the field if there's content or changes back to placeholder text if there's no content.

Fields with this style can also optionally show a small label below; this label is known as "supporting text", commonly used to display hints or error messages.

There are four new properties for the Entry-Field control to configure the Material Design layout:

MATERIAL-DESIGN = 1 or 2 to specify the boxed or underlined field style

MD-LABEL to specify the placeholder text that will become the floating label

MD-RADIUS to specify the radius for the box corners

MD-SUPPORTING-TEXT to specify the text of the optional label that is shown below the field

The following snippet:

```
03 Ef-mail Entry-Field
   line 3, col 3, size 30
   height-in-cells, width-in-cells
   material-design 1
   md-label "Email"
   md-supporting-text "Insert a valid email address"
   md-radius 50.
```

generates a field as depicted in Figure 7, *Boxed material design field*.

Figure 7. Boxed material design field.

Changing the value of the material-design property from 1 to 2:

```
03 Ef-mail Entry-Field
   line 3, col 3, size 30
   height-in-cells, width-in-cells
   material-design 2
   md-label "Email"
   md-supporting-text "Insert a valid email address"
   md-radius 50.
```

the field will look like the one depicted in Figure 8, *Underlined material design field*.

Figure 8. Underlined material design field.

The new property MD-SUPPORTING-TEXT specifies text to be displayed below the Entry-fields when the Material Design style is applied. This text can be either always visible (if set in Screen Section or via MODIFY statement before accepting user input) to provide users with a hint on how to fill the field or can be shown later (via MODIFY statement after a user action) to report an error.

Both MD-LABEL and MD-SUPPORTING-TEXT use the Entry-Field border color for the text color.

If you're using MD-SUPPORTING-TEXT to report an error, you might consider applying a red border color to the Entry-Field, making the label text red, as shown in the following code snippet:

```
perform check-email-address.
if not email-is-valid
   modify Ef-mail border-color 13
                  md-supporting-text "Invalid email address"
end-if.
```

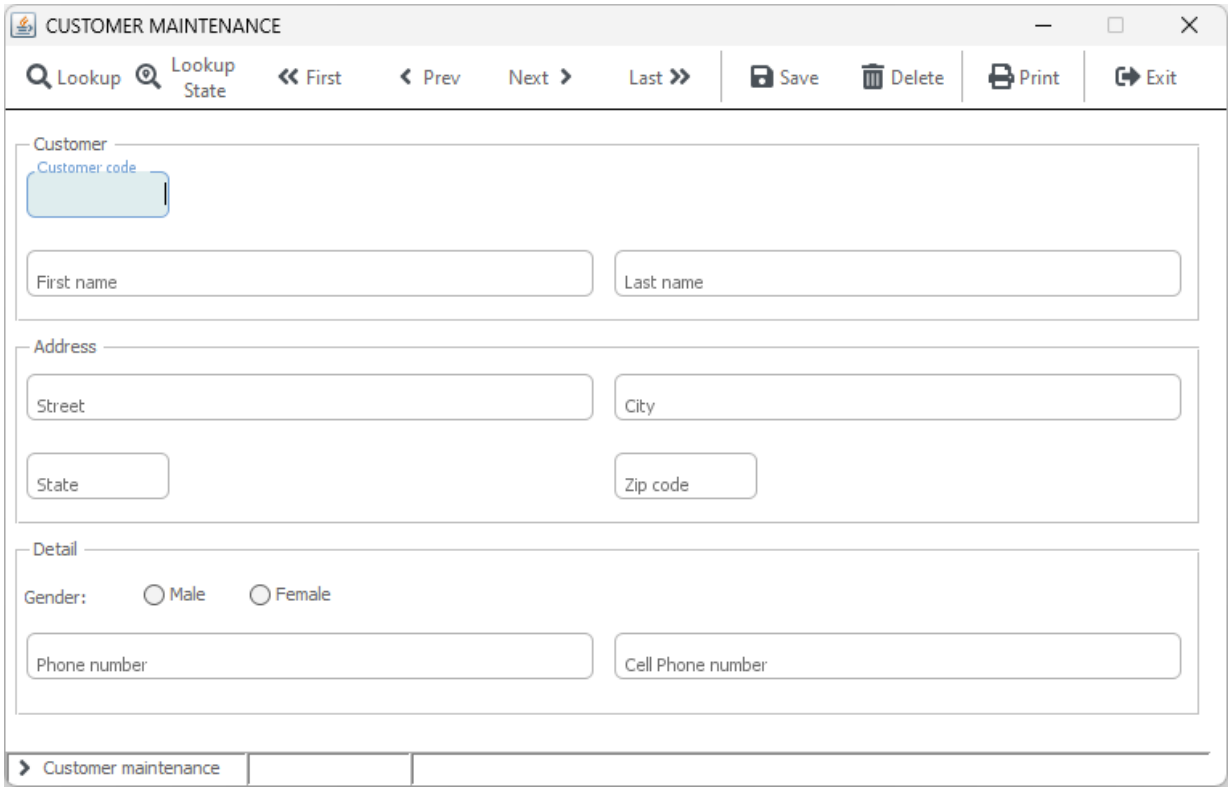
The above code, when the IF condition is true, will make the Entry-Field appear as depicted in Figure 9, *Error reported via border color and supporting text*.

Figure 9. Error reported via border color and supporting text.

When using the material-design property on all fields, jyou can design GUIs that do not specify label controls, since the labels are automatically provided by the entry field. An example of such user interface is installed with the samples, in the modernization\graphical\5-material-design-gui folder.

The result is shown in Figure 10, *Modernization sample with material-design*.

Figure 10. Modernization sample with material-design.



Better integration with LAF

The new style LAF-COLORS has been implemented on window, tool-bar and ribbon controls to simplify the management of colors for applications that wish to follow the default colors set by the Look and Feel.

Instead of calling the J$GETFROMLAF routine to retrieve the LAF background and foreground colors and then applying them in the DISPLAY WINDOW statement, you can now simply define the new style when creating the window. This ensures the window automatically uses the LAF colors, as shown in the following code snippet:

```
display standard window
        laf-colors
        ...
```

This style overrides any properties such as color, background-color, or foreground-color.

The style can be applied automatically to all windows across the application by adding the following compiler options and recompiling the source files:

```
iscobol.compiler.gui.window.defaults=laf-colors
iscobol.compiler.gui.tool_bar.defaults=laf-colors
iscobol.compiler.gui.ribbon.defaults=laf-colors
```

When applications adopt different LAFs, colors assigned directly to controls (such as push-buttons or entry-fields) or to their elements (such as grid cells, rows, or tree-view items) may need to be adjusted to better match the selected LAF, especially when using dark mode.

Two new op-codes have been implemented in the W$PALETTE library routine to help:

- WPALETTE-MAP-COLORS to map multiple RGB colors at once. With this function, all the colors used in the application can be remapped to different colors more suitable for the chosen LAF.
- WPALETTE-RESET-COLORS to reset all settings to the default values. With this function it's easier to manage on-the-fly LAF changes, for example when switching from Light to Dark mode or vice versa.

By leveraging these two op-codes, the runtime can efficiently remap or restore palette definitions at run time, enabling centralized management of multiple Look and Feel (LAF) configurations. As a result, all LAF-related logic is confined to the main program—or to the

module responsible for LAF switching—eliminating the need to refactor individual source modules that apply interface color attributes.

The syntax of the new functions is shown in the following code snippet:

```
call "W$PALETTE" using wpalette-map-colors
                       wpalette-original-colors
                       wpalette-new-colors
...
call "w$palette" using wpalette-reset-colors
```

The new parameters are defined in the updated copy file “ispalette.def” as shown below:

```
01  wpalette-original-colors.
    03  wpal-original-color occurs dynamic.
        05  wpal-original-red                  pic x comp-x.
        05  wpal-original-green                pic x comp-x.
        05  wpal-original-blue                 pic x comp-x.
01  wpalette-new-colors.
    03  wpal-new-color occurs dynamic.
        05  wpal-new-red                       pic x comp-x.
        05  wpal-new-green                     pic x comp-x.
        05  wpal-new-blue                      pic x comp-x.
```

Figure 11, *Colors used in Light mode*, shows an example of a window that displays two labels: one uses a blue text color, and another uses a gray text color on a black background, that are not optimal when using Dark mode, as shown in Figure 12, *Same colors used in Dark mode* . Also tool-bar icons that are dark gray in light mode are almost hidden in Dark mode. These issues can be solved easily using the new W$PALETTE op-codes and remapping the colors from blue to green, black to fuchsia and dark gray to white, as shown in Figure 13, *Remapped colors used in Dark mode*. This is how you can easily enable Dark mode in your gui application.

Figure 11. Colors used in Light mode.

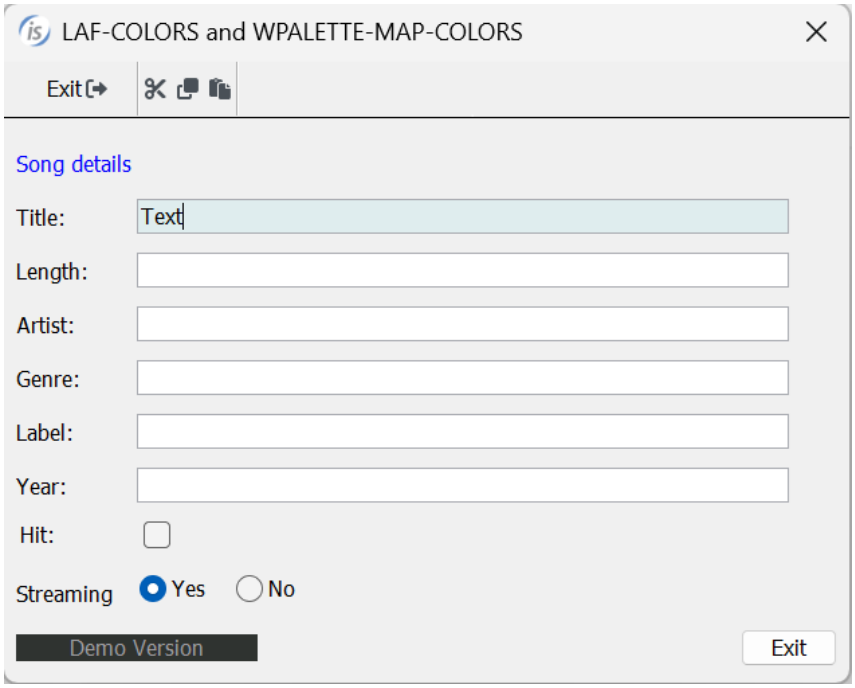


Figure 12. Same colors used in Dark mode.

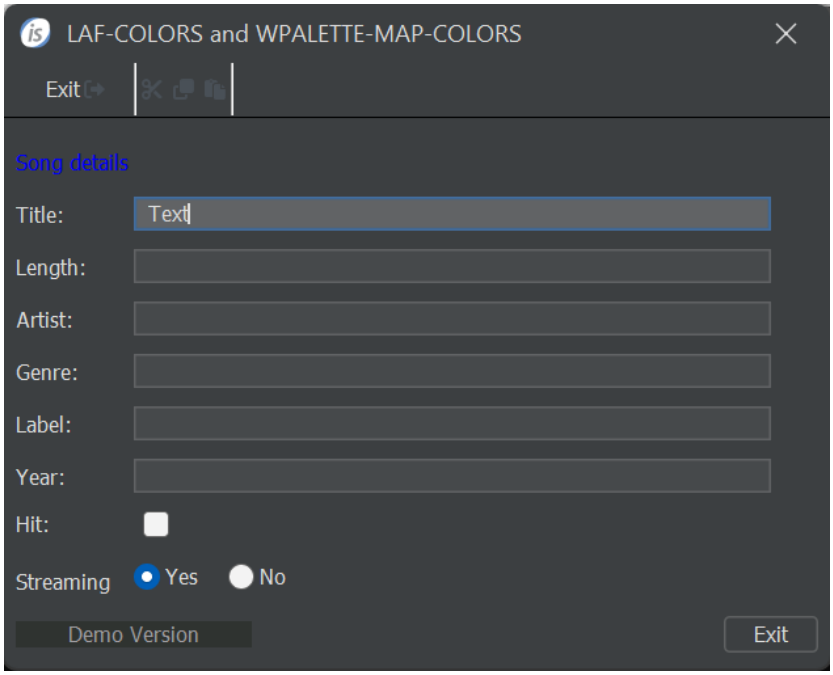
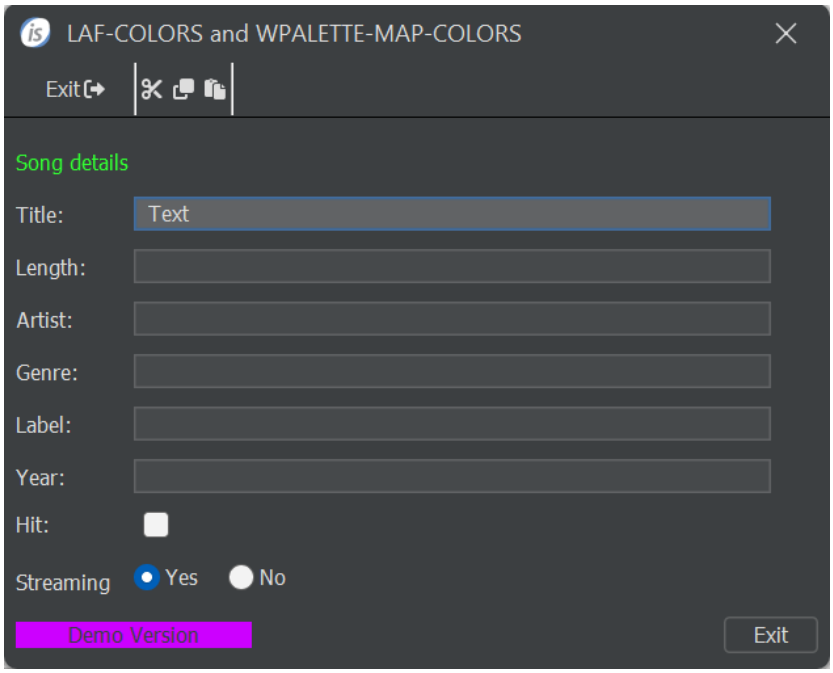


Figure 13. Remapped colors used in Dark mode.



In addition, colors and icons used by graphical messages created with DISPLAY MESSAGE BOX statement follow the defaults from the current Look and Feel. If the behavior of previous versions needs to be restored, you can set the configuration

```
iscobol.gui.messagebox.native_style=false
```

Other graphical improvements

Entry-field and combo-box controls have a new property named TRUNC-VALUE. This property can be used in INQUIRE statements to check if the content has been truncated because it's too large to fit in the control's width. This is especially useful when developing GUIs that support window resizing and need to adjust content to provide a responsive layout.

The following code snippet demonstrates one possible use case: if an entry-field content is truncated, the program sets the control hint to the full content, allowing users to hover over the control to view the entire value.

```
inquire ef-desc trunc-value in w-trunc
if w-trunc = 1
   inquire ef-desc value in w-desc
   modify ef-desc hint w-desc
end-if
```

BORDER-WIDTH and BORDER-COLOR properties are now supported in radio-button and check-box controls, as they have been supported by the PUSH-BUTTON control in previous releases. This is especially useful in controls that contain images that need to be underlined with the border.

These are new configurations options that have been implemented for GUI programs:

iscobol.gui.disabled_placeholder_color=n to set the color of disabled controls when the placeholder is visible.

iscobol.gui.secure_num_chars= n to display a fixed number of characters in secure fields to increase the user privacy and security.

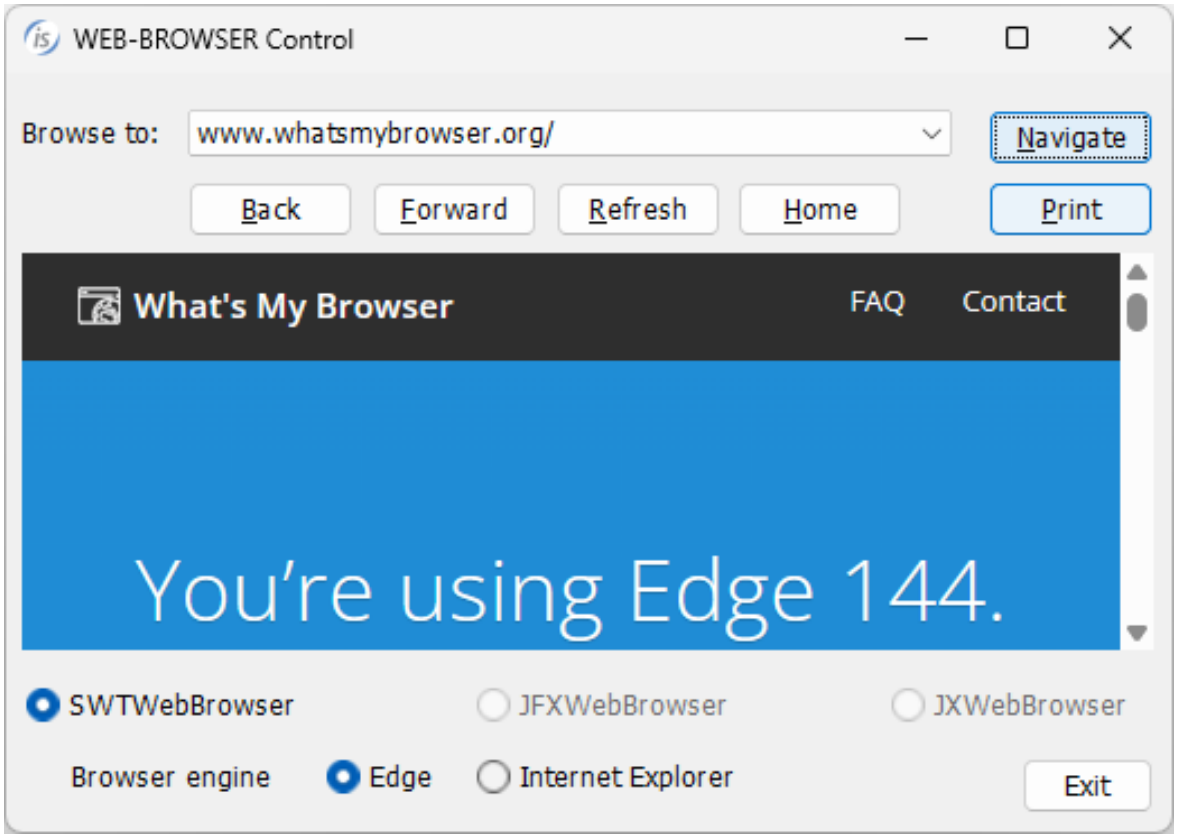
`iscobol.gui.secure_char`=x to set a different character used in secure fields, with the default value being "*".

The default value of the configuration option `iscobol.gui.webbrowser.class` has been changed to "`com.iscobol.browser.swt.SWTWebBrowser`". The previous browser implementation was based on the Microsoft Internet Explorer browser engine for Windows, and it has been deprecated in favor of the more modern Eclipse SWT Browser, based on Edge engine for Windows and WebKit2GTK for Unix.

If the COBOL application has been configured to use JavaFX's WebView, it will continue to use it.

The figure 14, *New web-browser SWT*, shows the new implementation, loading a site that returns which kind of browser is used.

Figure 14. New web-browser SWT.



isCOBOL Runtime

isCOBOL Evolve 2026 R1 provides updated versions of several third-party libraries to improve security and modernize applications. It also includes enhanced library routines.

Updated third party libraries

The Apache POI library, used by isCOBOL to create Excel-compatible .xlsx files—for example when exporting grid control content—has been upgraded in the isCOBOL 2026 R1 release from version 4.1.2 to version 5.5.1.

The previous iText library used to create pdf files, for example when creating printer files using "-P PDF filename.pdf", has been migrated to OpenPDF. PDF encryption is handled by the Bouncy Castle library, which has been updated from version 1.38 to version 1.78.1.

Keeping libraries up to date is essential to prevent bugs and security vulnerabilities, ensuring that applications remain stable, secure, and protected against newly discovered threats.

Library routines enhancements

A new library routine named C$PROGINSTACK has been implemented to check if a program is in the current stack. This can help programs better manage the execution of recursive calls. The library takes a program name as parameter, and the return code is 0 if the program is not in the stack or 1 if it is. The following code snippet shows the usage of the new routine:

```
move "pr-cust" to program-name
call "C$PROGINSTACK" using program-name
                     giving routine-status
if routine-status = 0
   ...
end-if
```

Maven integration

Apache Maven is one of the most used build automation and project management tools designed primarily for Java-based projects, though it also supports many other languages and frameworks. At its core, Maven simplifies the process of building, packaging, and deploying software by providing a standardized project structure and a declarative approach to configuration. Instead of relying on complex build scripts, developers describe their project using a Project Object Model (POM), an XML file that defines dependencies, plugins, build goals, and project metadata in a clear and consistent way. One of Maven's most significant contributions is its powerful dependency management system. Maven automatically downloads and manages libraries from remote repositories, ensuring that projects use the correct versions and that transitive dependencies are handled transparently. This reduces configuration overhead, minimizes version conflicts, and improves build reproducibility across different environments and teams.

With isCOBOL 2026 R1, the Maven build system can be seamlessly integrated into isCOBOL compilation tasks, enabling developers to adopt Maven for building COBOL applications.

For example, the following pom.xml file uses the isCOBOL Maven plugin to compile programs:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.test.isccdemo</groupId>
  <artifactId>iscc-test</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>iscc-test</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <repositories>
    <repository>
      <id>veryant-repo</id>
      <name>Private Maven Repo</name>
      <url>https://www.veryant.com/maven-repo</url>
    </repository>
  </repositories>
```

```
<pluginRepositories>
    <pluginRepository>
      <id>veryant-repo</id>
      <name>Private Maven Plugin Repo</name>
      <url>https://www.veryant.com/maven-repo</url>
    </pluginRepository>
  </pluginRepositories>

  <dependencies>
  </dependencies>

 <build>
  <plugins>
    <plugin>
      <groupId>com.iscobol</groupId>
      <artifactId>iscobol-maven-plugin</artifactId>
      <version>2026.1.1173.1</version>
  <dependencies>
    <dependency>
      <groupId>com.iscobol</groupId>
      <artifactId>iscobol</artifactId>
      <version>2026.1.1173.1</version>
    </dependency>
    <dependency>
       <groupId>org.apache.ant</groupId>
       <artifactId>ant</artifactId>
       <version>1.10.14</version>
    </dependency>
  </dependencies>
      <executions>
        <execution>
          <id>compile-cobol</id>
          <phase>compile</phase>
          <goals>
            <goal>compile</goal>
          </goals>
          <configuration>
           <destDir>${project.build.directory}/classes</destDir>
           <options>
             <option>-verbose</option>
           </options>
           <force>false</force>
           <filesets>
              <fileset>
                <directory>${basedir}/src</directory>
                <includes>
                  <include>**/*.cbl</include>
                </includes>
              </fileset>
           </filesets>
          </configuration>
        </execution>
      </executions>
    </plugin>

  </plugins>
</build>
</project>
```

By executing the Maven command:

```
mvn compile
```

all .cbl source files in the "src" folder are compiled and the .class files are created in the folder target\classes. When executed for the first time, Maven downloads the dependencies from the Veryant repository, providing the output shown below:

```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.test.isccdemo:iscc-test >----------
[INFO] Building iscc-test 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------
Downloading from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol-maven-plugin/2026.1.1173.1/iscobol-maven-plugin-
2026.1.1173.1.pom
Downloaded from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol-maven-plugin/2026.1.1173.1/iscobol-maven-plugin-
2026.1.1173.1.pom (484 B at 379 B/s)
Downloading from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol-maven-plugin/2026.1.1173.1/iscobol-maven-plugin-
2026.1.1173.1.jar
Downloaded from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol-maven-plugin/2026.1.1173.1/iscobol-maven-plugin-
2026.1.1173.1.jar (13 kB at 41 kB/s)
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ iscc-test
[INFO] skip non existing resourceDirectory:
\Veryant\isCOBOL_SDK2026R1\sample\build-automation\maven\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ iscc-test ---
[INFO] No sources to compile
[INFO]
[INFO] --- iscc:2026.1.1173.1:compile (compile-cobol) @ iscc-test --
Downloading from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol/2026.1.1173.1/iscobol-2026.1.1173.1.pom
Downloaded from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol/2026.1.1173.1/iscobol-2026.1.1173.1.pom
Downloading from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol/2026.1.1173.1/iscobol-2026.1.1173.1.jar
Downloaded from veryant-repo: https://www.veryant.com/maven-
repo/com/iscobol/iscobol/2026.1.1173.1/iscobol-2026.1.1173.1.jar
[INFO] isCOBOL Maven plugin Execution...true
[INFO] I --> C:\Veryant\isCOBOL_SDK2026R1\sample\build-
automation\maven\src\IO-INDEXED.cbl
[INFO] I --> C:\Veryant\isCOBOL_SDK2026R1\sample\build-
automation\maven\src\IO-LINESEQUENTIAL.cbl
[INFO] I --> C:\Veryant\isCOBOL_SDK2026R1\sample\build-
automation\maven\src\IO-PERFORMANCE.cbl
[INFO] I --> C:\Veryant\isCOBOL_SDK2026R1\sample\build-
automation\maven\src\IO-RELATIVE.cbl
[INFO] I --> C:\Veryant\isCOBOL_SDK2026R1\sample\build-
automation\maven\src\IO-SEQUENTIAL.cbl
[INFO] Compiling 5 source files
```

```
[INFO] ------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
[INFO] Total time:  10.937 s
[INFO] Finished at: 2026-01-26T11:11:52+01:00
[INFO] ------------------------------------------------------------
```

In successive runs, since the plugin is already installed and the dependencies have been downloaded, compilation occurs only if any of the source files have been modified. If no source files have changed, the output will be:

```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.test.isccdemo:iscc-test >-------------------
[INFO] Building iscc-test 1.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ iscc-test ---
[INFO] skip non existing resourceDirectory
C:\Veryant\isCOBOL_SDK2026R1\sample\build-utomation\maven\src\main\resources
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ iscc-test ---
[INFO] No sources to compile
[INFO]
[INFO] --- iscc:2026.1.1173.1:compile (compile-cobol) @ iscc-test ---
[INFO] isCOBOL Maven plugin Execution...true
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.649 s
[INFO] Finished at: 2026-01-26T11:29:58+01:00
[INFO] ------------------------------------------------------------------------
```

Improved compatibility

IsCOBOL 2026 R1 has been enhanced to improve compatibility with other COBOL dialects, such as MicroFocus COBOL and IBM COBOL. The compiler has been improved with new functions and ESQL statements.

Compatibility with other COBOLs

The ALLOCATE and FREE statements are supported to manage dynamic memory allocation and are also part of the latest ANSI 2023 standard. The allocated storage persists until explicitly released with a FREE statement, or when the run unit is terminated.

The following snippet of code shows how to use the new syntax to allocate and then free 1 KB of memory:

```
WORKING-STORAGE SECTION.
77  ptr     pointer.
LINKAGE SECTION.
01  lkitem  pic x(1024).
...
    allocate lkitem initialized returning ptr
    ...
    free ptr
```

In the variable declarations, new clauses are now supported:

- the USAGE DISPLAY-1 in PIC N defines an item as DBCS instead of Unicode UTF-16
- the USAGE NATIONAL clause in numeric and edited data items defines fields in UTF-16 like the standard PIC N
- the E usage clause defines External Floating-Point Data Type (EFP)

This is a code snippet that uses the new syntax:

```
    05  ws-nat         pic n(5)         usage national.
    05  ws-dbcs        pic n(5)         usage display-1.
    05  ws-num         pic 9(3)         usage national.
    05  ws-edit        pic z.zz9        usage national.
    05  ws-float       pic -9v9(9)E-99.
    05  ws-float-unic  pic +9v9(9)E+99  usage national.
```

A new intrinsic function named UID4 has been introduced. It returns a 36-character alphanumeric string representing a version 4 Universally Unique Identifier (UUID).

For example, the following code:

```
77  ws-uuid  pic x(36).
...
move function UUID4() to ws-uuid
```

will set the variable named ws-uuid to a string like "2eef112d-1f9b-408e-8c2a-22605e4d9e4d".

Compatibility with ESQL statements

The COMMENT ON statement adds a descriptive text to a database object (such as a table, column, view, or other schema element) and stores it in the system catalog for documentation and metadata purposes. Not all DBMS systems support comments, and those that do may require different syntaxes. The isCOBOL compiler does not perform a deep check on the syntax; it assumes that the developer used the format suitable for the targeted DBMS, generating a runtime error if not.

The snippet below shows how to add a comment on a table column in PostgreSQL:

```
EXEC SQL
   COMMENT ON COLUMN my_table.my_column
           IS 'Employee ID number'
END-EXEC.
```

This other snippet shows how to add a comment on multiple columns at once on IBM DB2:

```
EXEC SQL
   COMMENT ON DSN8C10.DEPT
      (MGRNO IS
       'EMPLOYEE NUMBER OF DEPARTMENT MANAGER',
       ADMRDEPT IS
       'DEPARTMENT NUMBER OF ADMINISTERING DEPARTMENT');
END-EXEC.
```

Improved Debugger

IsCOBOL 2026 R1 debugger can now set breakpoints while the program is running.

With previous versions, and in many other COBOL debuggers, new breakpoints can only be set when programs are not running.

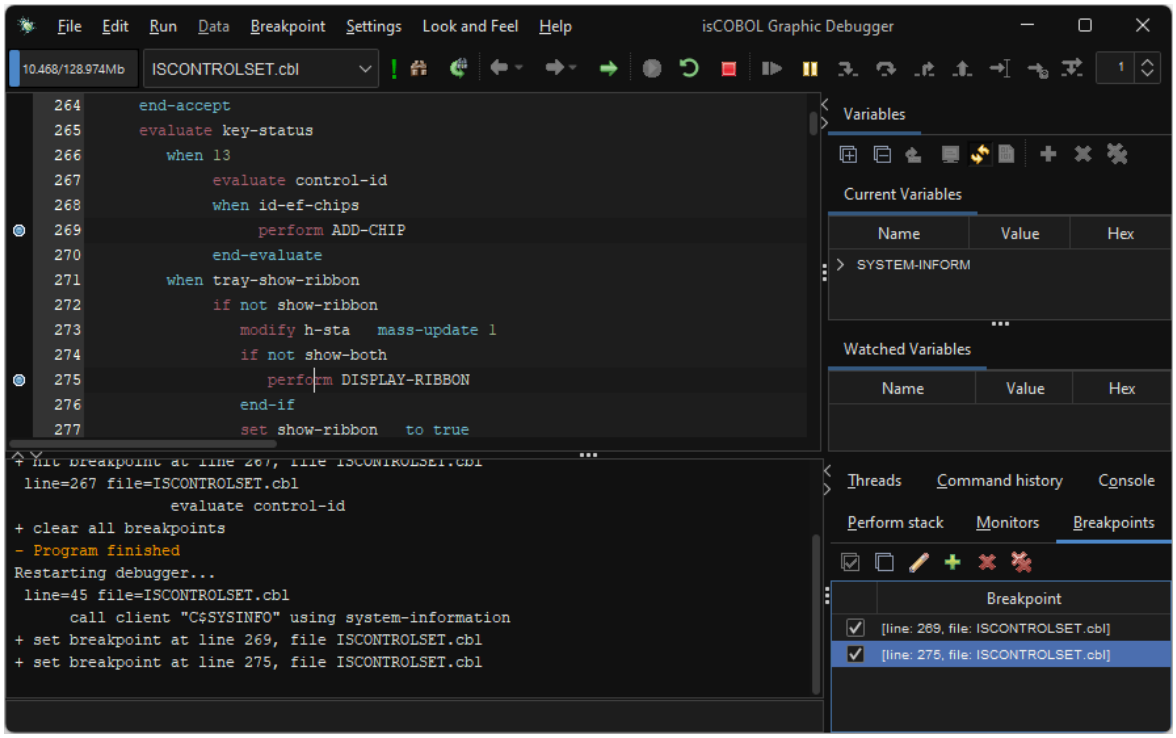
Starting from 2026 R1, the isCOBOL Debugger's "Set breakpoint" menu items are always enabled, even when programs are running.

This new feature is particularly useful when running multiple programs, some compiled in debug mode and some are not. Even if a long-running task is being executed in a program not compiled with debug information, the debugger still allows developers to set breakpoints without having to wait for a debug-mode program to take control.

Breakpoints can also be set by clicking on the header column at the left of the source line or using the `BREAK` command in the command-line area.

The new feature is shown in Figure 15, *Breakpoints set in Debugger while program is running*, where two breakpoints are set on lines 269 and 275.

Figure 15. Breakpoints set in Debugger while program is running.

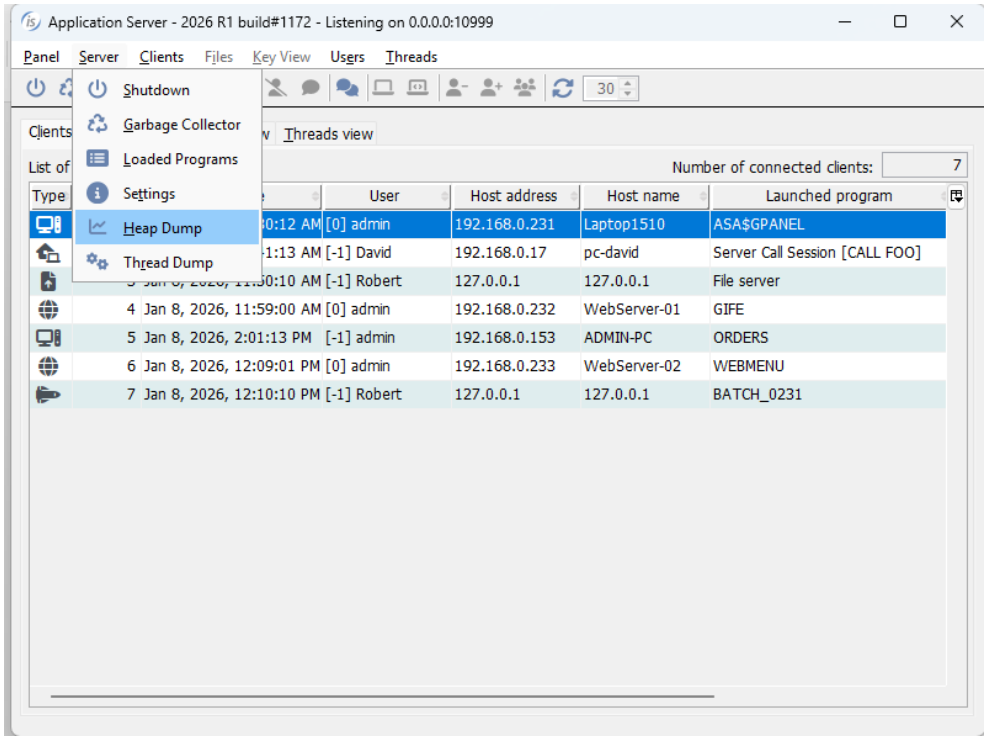


Improved isCOBOL Server

IsCOBOL 2026 R1 can now generate thread dumps and heap dumps directly from the isCOBOL Server Panel.

The isCOBOL Server Panel includes two new menu items in the Server menu, along with two corresponding toolbar buttons. These features allow you to collect heap memory dumps and thread dumps directly from the Panel, without needing to attach the isCOBOL Server to external diagnostic tools such as VisualVM. The picture in Figure 16, *Panel dump features*, shows the new items in the Server menu.

Figure 16. Panel dump features



The dumps are generated on the server, in the isCOBOL Server's working directory, and a message box will be displayed showing the full file name.

IsCOBOL WebClient

IsCOBOL WebClient adds a new window style named IWC-DOCKABLE to mark windows as dockable in web environments.

WebClient lets users undock windows using a dedicated button on the window title bar that the user can click.

In previous releases, this setting affected all windows: they were either all dockable or all undockable. Now each window can be set to be dockable by using the new IWC-DOCKABLE style. The following snippet creates an independent window marked as dockable:

```
display independent graphical window
        iwc-dockable
        ...
```

Figure 17, *Docking mode setting,* shows the values available to the Docking Mode setting. When set to ALL, all windows are automatically dockable. When using MARKED, only windows that have new IWC-DOCKABLE will be dockable, and when set to NONE, no windows will be dockable, regardless of windows styles.

Figure 17. Docking Mode setting

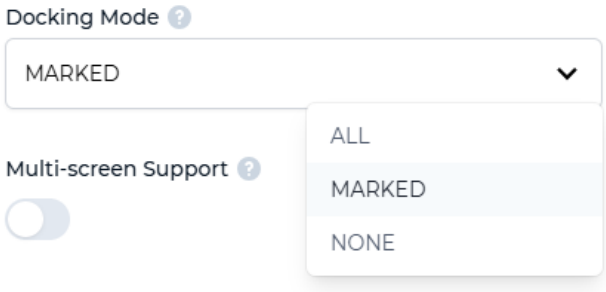


Figure 18, *Undockable subwindow in WebClient*, and Figure 19, *Undocked subwindow in WebClient*, show two independent windows, a parent and child, where the child can be undocked, and the parent can't. The second image shows how the child window appears after clicking the undock button on its title bar: it is moved in a separate browser window.

Figure 18. Undockable subwindow in WebClient.

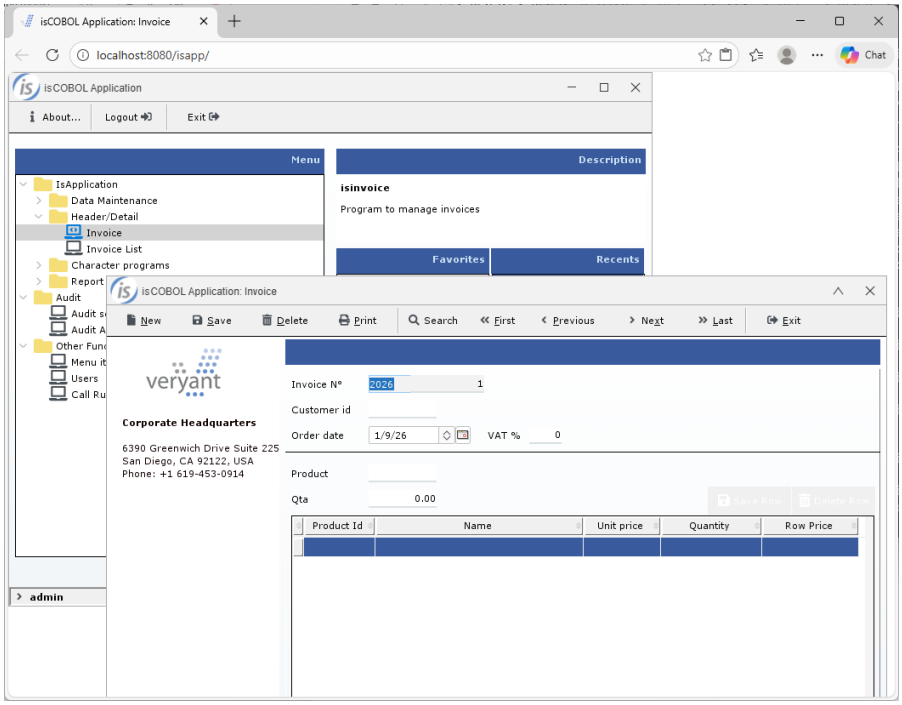
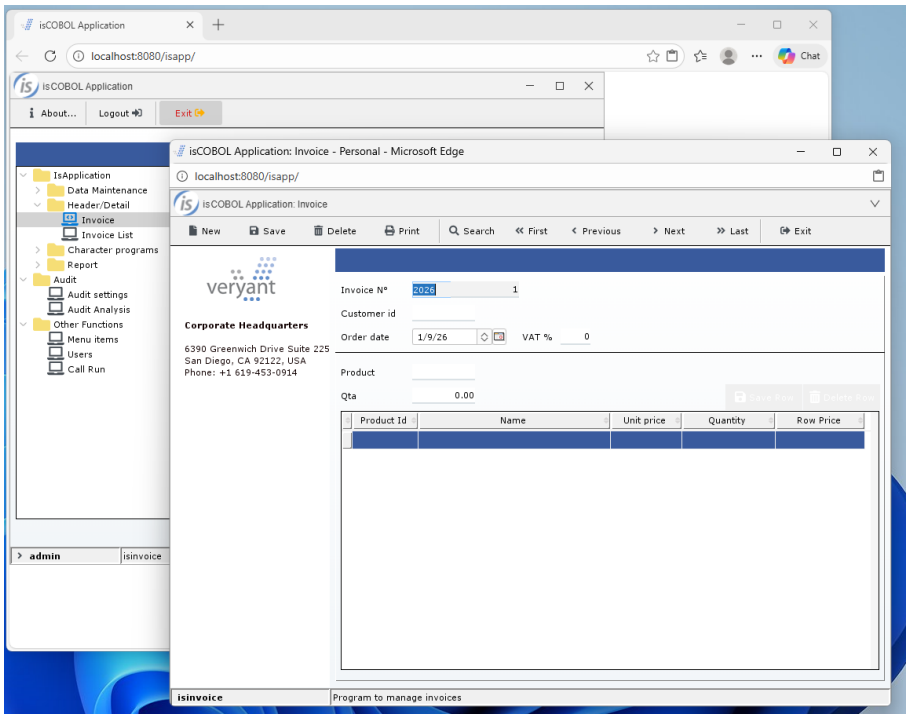


Figure 19. Undocked subwindow in WebClient.



The default docking behavior is moving the window to a new browser window, but it can be configured in the Docking Type setting, and can be set to have the window moved to a separate tab of the same browser window, as depicted in Figure 20, *Docking Type Setting*.

Figure 20. Docking Type setting.

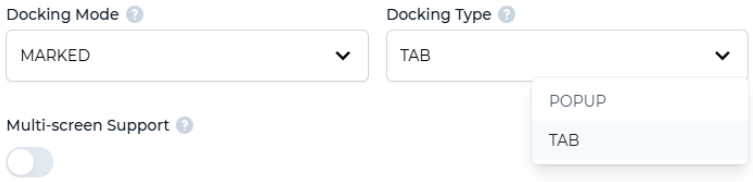
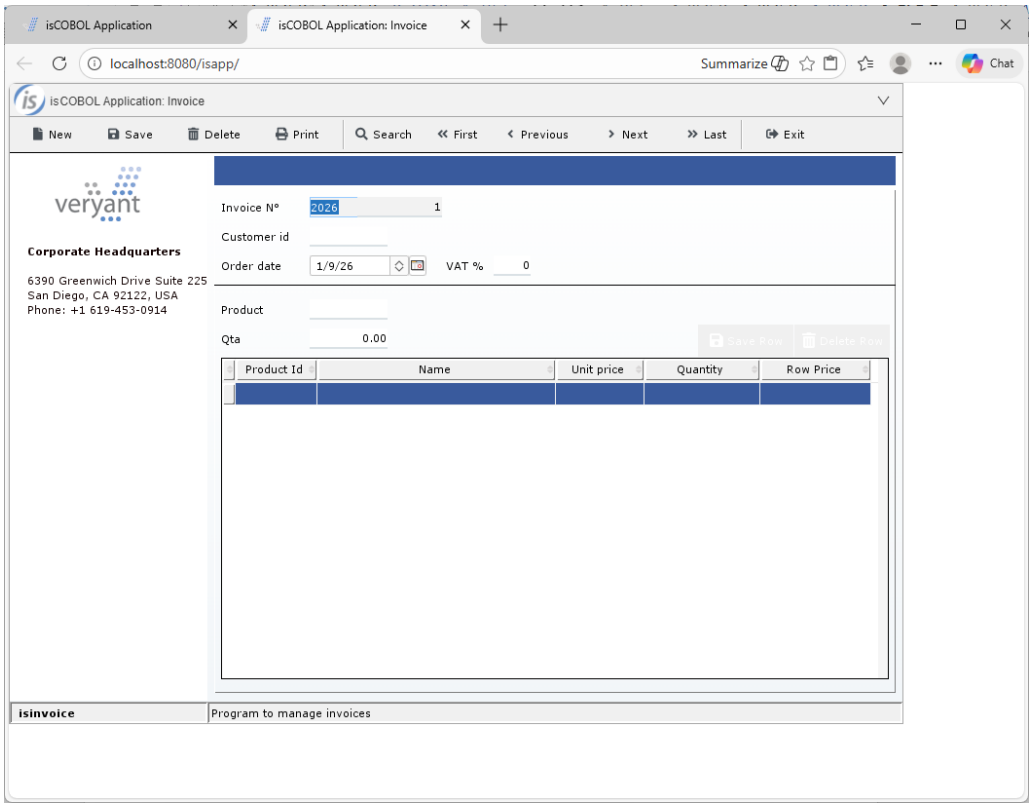


Figure 21, *Subwindow undocked into separate tab*, shows how the previous undocked window appears in a separate browser tab after changing the Docking Type setting.

Figure 21, *Subwindow undocked into separate tab*.



Database Bridge

Database Bridge 2026 R1 offers two new properties that can be useful when migrating an application from ISAM files to Oracle RDBMS.

A traditional COBOL application that uses ISAM files may have the following directory structure:

```
/app/data/2024/invoices.dat
/app/data/2025/invoices.dat
/app/data/2026/invoices.dat
```

This provides a clear separation of data across years. To carry this concept to an RDBMS environment, the Veryant Database Bridge can automatically create different table names, and you can specify what part of the folder structure to apply to the naming, using the iscobol.easydb.dirlevel runtime configuration property.

This property sets the number of path segments of the file name, starting from the bottom of the hierarchy and going to the top, to use when naming tables.

For example, by setting iscobol.easydb.dirlevel=1, the data of invoices.dat file in the /app/data/2024 folder is stored in a table named 2024invoices, the data of invoices.dat in /app/data/2025 is stored in a table named 2025invoices and the data of invoices.dat in /app/data/2026 is stored in a table named 2026invoices.

This solution is available for all the RDBMS supported by isCOBOL Database Bridge.

Starting from version 2026 R1, and only for the Oracle RDBMS, an alternative solution lets you store all data from the multiple invoices files in a single invoice table, and Database Bridge creates an additional varchar column called DTC , to store the folder information.

The DTC column is added at the beginning of the primary key and every index. This is achieved by setting two configuration properties:

`iscobol.compiler.easydb.dirlevel_to_columns=true` to enable the feature in the Oracle EDBI routines when the compiler generates the EDBI class

`iscobol.easydb.dirlevel_to_columns=n` to specify at runtime the number of path segments to store in the column

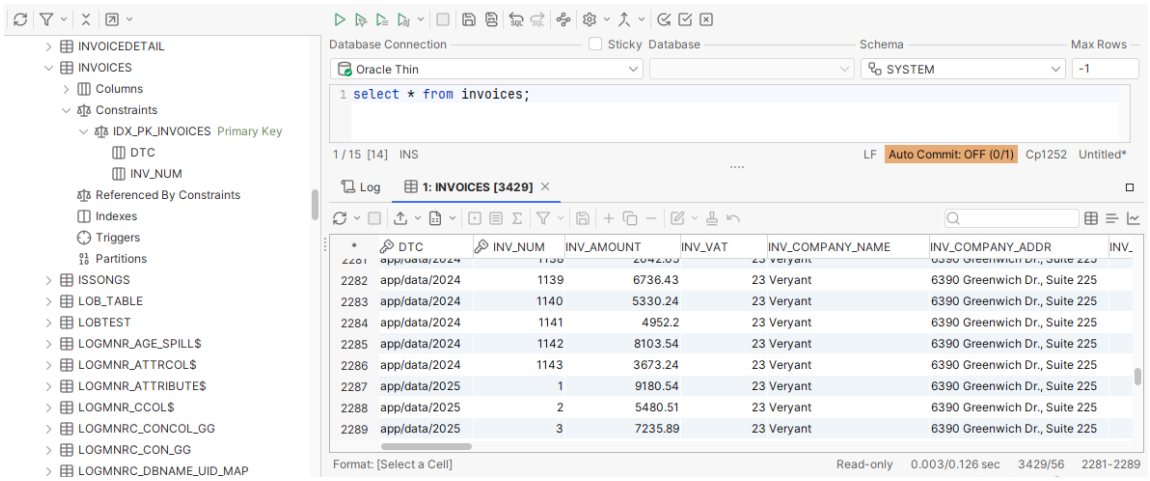
In the example above, using the following configuration settings

```
iscobol.compiler.easydb.dirlevel_to_columns=true
iscobol.easydb.dirlevel_to_columns=3
```

Results in the creation of a table called “invoices” with the same structure of the ISAM file, and an additional column named DTC. All records of invoices.dat from /app/data/2024 will have “app/data/2024” as value of the DTC column, all records of invoices.dat from /app/data/2025 will have “app/data/2025” as value of the DTC column, and all records of invoices.dat from /app/data/2026 will have “app/data/2026” as value of the DTC column.

With this configuration, when using external programs that query the table through SQL, all data from the various files can be retrieved at once or filtered by specifying the desired values in the DTC column. Figure 22, *Invoices table with the DTC column,* below shows how data from different invoice files was stored in the same invoice table and the DTC column contains the path of the file that the COBOL application intended to use.

Figure 22. Invoices table with the DTC column



c-treeRTG v5

The c-treeRTG product has been updated to version 5. This new version contains improvements and optimizations in several areas, such as SQL access, GUI tools, and web-based tools. It also features updated python and Direct SQL API for C/C++ drivers and libraries.

Useful new features for isCOBOL users include the ability to increase the record length of an indexed file and add new keys without needing to migrate existing records.

These operations can be executed using the ctutil command line utility or with a new isCOBOL configuration setting that is used when opening files using ctreej.

New ctutil options

`-alter` to change the record schema of an existing file using .iss file definitions

`-makeidx` to add new indices to an existing file using .iss file definitions

`-augment` to increase the record length

The first 2 options require the .iss file created by the isCOBOL compiler with the –efc compiler option and can be used on indexed files that are already sql-ized, making the data accessible using SQL.

The augment option can be used on all C-Tree indexed files, even if they are accessed using COBOL's legacy ISAM access (OPEN, READ, WRITE, …).

For example, if an existing C-Tree tree file has a fixed record length of 1600 bytes, running the following command:

```
ctutil -augment customers 2048
```

The output will be:

```
ctutil
Version 5.0.3.173-250510 - isCOBOL Edition
Record length           :         1600 bytes
 Variable               :          yes
 Max length             :         2048 bytes

Operation completed successfully.
```

The indexed file named customers now has a variable record length with minimum size 1600 bytes and maximum size 2048.

New configuration for ctreej

The new configuration `iscobol.ctree.auto_adjust=true` is supported by the ctreej interface and forces the record size defined by the program before the file is opened, ensuring that a record size mismatch can never occur. All programs that use the file need to be recompiled using the updated FD definition. Also new keys are automatically added to match the file description used to open the file.

Details on the changes are traced in the runtime log, if enabled. The property affects only the OPEN I-O statements.

With this new configuration, there is no longer a need to write dedicated programs to migrate data from the old file definition to the new one. The time required to run such migration programs is also eliminated, as the file is automatically adjusted at runtime when the OPEN I-O statement is executed.

For example, if an existing indexed file has this COBOL definition:

```
select cust assign to "customers"
       organization is indexed access is dynamic
       record key is cust-key
       alternate record key is cust-name.
...
fd  cust.
01  rec-cust.
    03 cust-key      pic 9(8).
    03 cust-name     pic x(100).
    03 cust-date     pic 9(8).
    03 cust-address  pic x(200).
    03 cust-mail     pic x(100).
    03 cust-contact  pic x(160).
    03 cust-notes    pic x(1024).
```

the ctree file has a fixed record length of 1600 bytes and two keys.

If additional fields and keys are required, the COBOL definitions can be modified, for example:

```
select cust assign to "customers"
       organization is indexed access is dynamic
       record key is cust-key
       alternate record key is cust-name
       alternate record key is cust-address with duplicates.
...
```

```
fd  cust RECORD VARYING IN SIZE FROM 1600 TO 2048.
01  rec-cust.
    ...
    03 cust-notes    pic x(1024). |this was the previous last field
    03 cust-new-f1   pic x(200).
    03 cust-new-f2   pic x(248).
```

As a result two new fields are appended to the record, increasing the record length by 448 bytes, and a new key on the cust-address field is added to the existing keys.

If the program with the new definition is compiled and executed without the new configuration, the OPEN statement fails, returning the expected File status code 39,02 since the record length mismatch. If the program is executed using the following configuration settings:

```
iscobol.file.index=ctreej
iscobol.ctree.auto_adjust=true
```

the OPEN I-O statement automatically adjusts the file definitions, and the program can read existing records and write new records with the new additional fields.

Running the “ctutil –info customers” command from the command line will show the new record structure, as shown below.

```
ctutil
Version 5.0.3.173-250510 - isCOBOL Edition

File name               :    customers
...
Record length           :         1600 bytes
 Variable               :          yes
 Max length             :         2048 bytes
...
Number of indices       :            3
 Key# Size Dups Null Mod Compress Segments
   1    8   no   no  yes    no       1
   2  100   no   no  yes    no       1
   3  200  yes   no  yes    no       1
Alt. Collating Sequence :           no

Operation completed successfully.
```