isCOBOL Evolve: Introduction

Key Topics:

- Introduction to isCOBOL™ Evolve
- The Basics
- Wrappers
- Special Features

Chapter 1

Introduction to isCOBOL™ Evolve

Introduction to isCOBOL™ Evolve

Congratulations on your selection of isCOBOL Evolve. This document will show you how to download and install isCOBOL Evolve products, and compile and run a simple application. It also includes an introduction to key terms and concepts related to the Java environment that are necessary to understand when using isCOBOL Evolve products.

isCOBOL Evolve is a complete environment for COBOL application development and deployment that offers the following components:

- isCOBOL Compiler – a Java-based 100% portable COBOL compiler supporting the latest ANSI standards and common legacy dialects. The isCOBOL Compiler also includes support for ESQL, Object Oriented COBOL, Unicode, and JavaBean graphical controls. The isCOBOL Compiler comes with the following utilities:
    - ISL - creates startup commands for your COBOL programs
    - ISMIGRATE – migrates ISAM to ISAM, ISAM to RDBMS, RDBMS to RDBMS, or RDBMS to ISAM.
    - GIFE – graphical indexed file editor.
    - ISCONFIG – converts a runtime configuration file from other COBOL systems to an isCOBOL properties file.
    - STREAM2WRK – creates a COBOL record description from XML, JSON and WSDL files.
    - COBFILEIO – generates classes for accessing COBOL files and records from Java programs.
    - JDBC2FD – generates FD (file description) copybooks from database tables.
    - ISSORT - sort indexed, sequential and relative files.
- isCOBOL Runtime Framework – a single, easily deployed file, written completely in Java, that implements COBOL verbs. The isCOBOL Runtime Framework enables applications to run on any device supporting a Java Virtual Machine (JVM) -- from mainframes to mobile phones -- including application logic, user interface, and data access.
- isCOBOL Debugger – a 100% portable, graphical, source-level COBOL debugger that includes support for remote debugging.
- isCOBOL IDE – an adaptable, graphical, Integrated Development Environment (IDE) for all development tasks (design, coding, testing, debugging).

- isCOBOL Database Bridge – a Rapid Application Development (RAD) tool that analyzes a COBOL File Description (FD) and generates a file interface program with embedded SQL in COBOL. When you run, the COBOL File I/O operations on this data file are routed through the generated COBOL/ESQL interface which accesses the RDBMS via JDBC. (This product requires a special license to use with the isCOBOL Runtime Framework.)
- isCOBOL Application Server – an application server which enables high performance, multithreaded thin client processing and distributed processing. Since the thin client is Java based, you leverage the same GUI, regardless of the client platform (Windows, UNIX, Macintosh, handheld, etc.)
- isCOBOL JISAM – a Java-based indexed file system that works on a range of Java compatible platforms from mainframes to handheld devices. This product includes the JUTIL utility for basic file management.
- isCOBOL EIS – an umbrella of tools and features that allows development and execution of a web based application in a J2EE container. It includes Web Direct 2.0, a Web 2.0 solution which utilizes Asynchronous Javascript and XML (AJAX) technology to enable quick deployment of fully-interactive COBOL programs for the Web without changing basic application structure.
- c-treeRTG – a fast and secure client/server indexed file system.
- c-treeRTG SQL features – an RDBMS that uses c-tree Server files as the data store thus eliminating the need for data migration and providing an SQL interface to c-tree files through JDBC, ODBC, ADO.NET, PHP, PYTHON and SQL utilities.
- isCOBOL File Server - a file server which enables remote files handling.
- isCOBOL UDBC - a data management system that allows to work on indexed files hosted by the isCOBOL File Server as if they were a RDBMS.
- isCOBOL Mobile - a comprehensive, cost-effective COBOL solution for application development on mobile devices.
- JOE - a polymorphic script environment.

System Requirements

isCOBOL SDK System Requirements

Windows

- Windows 10 (using JDK 7u85 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 SP2 and 2008 R2 SP1 (64-bit)
- Windows Server 2012 (64-bit) and 2012 R2 (64-bit)
- RAM: 128 MB; 64 MB for Windows XP (32-bit)
- Disk space: 256 MB
- JDK 1.7

Note: As of April 8, 2014 Microsoft stopped supporting Windows XP and therefore it is no longer an officially supported platform. Users may still continue to use isCOBOL on Windows XP at their own risk, but support will only be provided against Microsoft Windows releases Windows Vista or later.

Mac OS X

- Intel-based Mac running Mac OS X 10.7.3 (Lion) or later.

- Administrator privileges for installation
- 64-bit browser [A]
- JDK 1.7

[A]A 64-bit browser (Safari, for example) is required to run Oracle Java on Mac.

Linux

- Oracle Linux 5.5+
- Oracle Linux 6.x (32-bit), 6.x (64-bit)
- Oracle Linux 7.x (64-bit) (7u67 and above)
- Red Hat Enterprise Linux 5.5+, 6.x (32-bit), 6.x (64-bit)
- Red Hat Enterprise Linux 7.x (64-bit) (7u67 and above)
- Suse Linux Enterprise Server 10 SP2, 11.x
- Suse Linux Enterprise Server 12.x (7u75 and above)
- Ubuntu Linux 10.04 and above
- JDK 1.7

isCOBOL IDE System Requirements

Windows

- Windows 10 (using JDK 7u85 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 SP2 and 2008 R2 SP1 (64-bit)
- Windows Server 2012 (64-bit) and 2012 R2 (64-bit)
- RAM: 512 MB;
- Disk space: 500 MB
- JDK 1.7

Note: As of April 8, 2014 Microsoft stopped supporting Windows XP and therefore it is no longer an officially supported platform. Users may still continue to use isCOBOL IDE on Windows XP at their own risk, but support will only be provided against Microsoft Windows releases Windows Vista or later.

Mac OS X (64 bit)

- Intel-based Mac running Mac OS X 10.7.3 (Lion) or later.
- Administrator privileges for installation
- 64-bit browser [A]
- JDK 1.7

[A]A 64-bit browser (Safari, for example) is required to run Oracle Java on Mac.

Linux (64 bit)

- Oracle Linux 5.5+

- Oracle Linux 6.x 6.x
- Oracle Linux 7.x 7u67 and above)
- Red Hat Enterprise Linux 5.5+, 6.x 6.x
- Red Hat Enterprise Linux 7.x 7u67 and above)
- Suse Linux Enterprise Server 10 SP2, 11.x
- Suse Linux Enterprise Server 12.x (7u75 and above)
- Ubuntu Linux 10.04 and above
- JDK 1.7

Chapter 2

The Basics

isCOBOL Evolve includes several pieces of software that are basic to any software development environment. The following pieces of software form the core of isCOBOL Evolve:

isCOBOL Evolve Product	Command to invoke the product
isCOBOL Compiler	iscc
isCOBOL Runtime Framework	iscrun
isCOBOL Debugger	iscrun -d
isCOBOL IDE	isIDE.exe

Notice that the isCOBOL compiler, runtime framework and debugger are all contained in a single file, "iscobol.jar", placed in the isCOBOL lib directory. This file is discussed below.

isCOBOL and Java

isCOBOL Evolve is tightly integrated with Java technology. In a nutshell, the isCOBOL Compiler translates COBOL source code into Java classes that are executed with the Java Virtual Machine (JVM).

Developers continue normal COBOL programming with isCOBOL Evolve. Behind-the-scenes, the isCOBOL Compiler translates COBOL source code into Java source code every time you compile. The isCOBOL Compiler feeds this Java source code to the Java compiler, which produces Java bytecode objects which can be executed with a JVM. This process is transparent for developers.

NOTE - Although the Java source code is temporary intermediate output, you can compile with a special “-jj” option if you ever want to retain this code.

The JDK and JRE

isCOBOL Evolve requires a Java Development Kit (JDK) for development and a Java Runtime Environment (JRE) for deployment.

Required to compile a COBOL program	Required to run a COBOL program
isCOBOL Compiler	isCOBOL Runtime Framework
Java Development Kit (JDK)	Java Runtime Environment (JRE)

NOTE - The JDK and JRE are not distributed with isCOBOL Evolve. They must be installed before using isCOBOL Evolve. For most platforms, they can be downloaded free of charge from "http://www.oracle.com/technetwork/java/javase/downloads/index.html". For more information, refer to the installation section.

The minimum Java version required is 1.7. Previous versions are not supported.

The JDK contains the Java compiler, utilities and a copy of the JRE.

On Microsoft Windows® platforms, the JDK and JRE are installed in subdirectories of "C:\Program Files\Java". For example, the default installation directory for JDK 7 is "C:\Program Files\Java\jdk1.7.0".

The default installation directory for the JRE that comes with JDK 7 is "C:\Program Files\Java\jdk1.7.0\jre".

The JRE is distributed separately for production-only sites where no development kit is needed. The default installation directory for JRE 7 is "C:\Program Files\Java\jre1.7.0".

The Java Virtual Machine (JVM)

The JRE contains the JVM which executes Java bytecode objects such as those produced by the isCOBOL Compiler. The JVM is native binary executable machine code. It is located in one or more Dynamic Link Libraries (DLLs) on Windows or shared libraries on UNIX.

The JRE provides an executable named "java" that uses the JVM to execute a main program. After compiling with the isCOBOL Compiler, you can execute a COBOL main program named MAINPROG with "java MAINPROG". (The name must be in all uppercase because Java class names are case sensitive)

NOTE - The JVM DLL or shared object can also be used directly by other executables such as a Java Enterprise Edition (Java EE) server or a transaction processor.

Java Classes

Since the Java programming language is object-oriented, code is organized using object classes. The bytecode objects produced by the Java compiler are called "class files", or simply "classes", and are named using the .class file name extension.

The isCOBOL Compiler, isCOBOL Runtime Framework, and isCOBOL Debugger are distributed as Java classes. After compiling a COBOL program with the isCOBOL Compiler, the COBOL object files are Java class files.

JAR Files and the Jar Utility

A Java Archive (JAR) file is an archive containing one or more files or directories, similar to a UNIX tar file or a Zip file. JAR files primarily contain Java class files, but any type of file can be included. These files use the file name extension ".jar".

Jar is also the name of the utility program that is used to create, update, list contents, and extract contents from JAR files. The jar utility is included in the JDK and its usage is similar to the UNIX tar utility.

Usage examples:

1. Create a jar library named *myApp.jar* putting all the classes of your project into it:

```
cd /develop/myApp/output
jar -cf myApp.jar *.class
```

2. Update *myApp.jar* by replacing MENU.class that has just been modified and recompiled:

```
cd /develop/myApp/output
jar -uf myApp.jar MENU.class
```

3. List the classes contained in myApp.jar by displaying their name on the system output:

```
jar -tf myApp.jar
```

Note - The content of a jar can also be viewed with a graphical interface if you open the jar file using an archive manager like WinZip, WinRar or 7Zip.

Class loading

In order for the JVM to find Java class files, they must be located in a directory contained in the class path, or stored in a JAR file that is listed in the class path. The Java compiler also uses the class path to locate classes referenced by the source file it is compiling.

The class path is most commonly specified in an environment variable named CLASSPATH. Other Java utilities, such as javap, also use the class path.

On Windows, the class path is a semicolon-delimited list of directories and/or JAR files, similar to the format of the PATH variable, except that in addition to directories, you can specify JAR files. The Java class loader treats JAR files just like directories.

Starting with Java version 6 (also known as 6.0 and 1.6.0) class path entries can contain the basename wildcard character *, which is considered equivalent to specifying a list of all the files in the directory with the extension .jar or .JAR. See below for links to additional information.

For example:

```
SET CLASSPATH=.;C:\Program Files\Veryant\isCOBOL2018R1\lib\*;
C:\Program Files\Java\jdk1.7.0\lib\tools.jar
```

The file names in bold in this example are explained below.

The JDK **tools.jar** File

The isCOBOL Compiler calls the Java compiler via its programmatic interface, not by executing a separate javac process. So in order to compile a COBOL program, the class path must contain the tools.jar file (as shown above). This file contains the programmatic interface to the Java compiler which is a Java class named “com.sun.tools.javac.Main”.

The JRE **rt.jar** File

In order to run a COBOL program, the class path must contain the rt.jar file (as shown above). This file contains the Java runtime classes.

NOTE - The latest versions of the Java runtime automatically add rt.jar to the class path.

The **iscobol.jar** File

The iscobol.jar file contains Java classes used for compiling, running and debugging a COBOL program. For example, in iscobol.jar the isCOBOL Compiler main class is named "com.iscobol.compiler.Pcc", and the debugger main class is named "com.iscobol.debugger.GraphDebugger". So if iscobol.jar is in the class path, you can execute the compiler with "java com.iscobol.compiler.Pcc", and the debugger with "java com.iscobol.debugger.GraphDebugger".

The isCOBOL Compiler uses classes that are in the JDK tools.jar and JRE rt.jar files.

The isCOBOL Runtime Framework and isCOBOL Debugger use classes that are in the JRE rt.jar file. They do not require the JDK tools.jar.

To use iscobol.jar:

1. Make sure that iscobol.jar is listed in the class path setting
2. Include the JDK tools.jar and/or JRE rt.jar if necessary in the classpath setting

For example (the following examples have single lines that may be broken up by print formatting),

```
SET CLASSPATH=.;C:\Program Files\Veryant\isCOBOL2018R1\lib\*;
C:\Program Files\Java\jdk1.7.0\lib\tools.jar
java [options] class_name
```

or

```
java -cp=".;C:\Program Files\Veryant\isCOBOL2018R1\lib\*;
C:\Program Files\Java\jdk1.7.0\lib\tools.jar" [options] class_name
```

COBOL programs classes

Unlike the libraries that were previously mentioned, COBOL programs classes can be loaded either from the class path or from the code prefix. It depends by the configuration property `iscobol.code_prefix`.

If such property is set, then COBOL programs classes are searched among the paths specified by the property, otherwise they are searched in the class path. However, if the same class is found in both code prefix paths and class path, then the class is loaded from the class path, hence it's not good practice to add the same path to both class path and code prefix.

Classes loaded from the code prefix can be cancelled and reloaded multiple times during the runtime session. Instead, classes loaded from the class path are loaded once and then kept in memory for the whole runtime session.

The Library Path

In order for the JVM to find shared libraries and shared objects, they must be located in a directory contained in the library path.

The JVM looks for the library path by inquiring a system dependent environment variable. The below table shows the most common ones:

Operating System	Library file extension	Environment Variable
AIX	*.so	LIBPATH
HP-UX Itanium	*.so	LD_LIBRARY_PATH
HP-UX PA-RISC	*.sl	SHLIB_PATH
Linux	*.so	LD_LIBRARY_PATH
Mac OS X	*.jnlib	DYLD_LIBRARY_PATH
SCO	*.so	LD_LIBRARY_PATH
Solaris	*.so	LD_LIBRARY_PATH
Windows	*.dll	PATH

More Information

For more information about these Java concepts, here are some links to helpful articles available from Wikipedia and Oracle.

Setting the class path:

http://en.wikipedia.org/wiki/Classpath_(Java)

http://download.oracle.com/javase/6/docs/technotes/tools/solaris/classpath.html

http://download.oracle.com/javase/6/docs/technotes/tools/windows/classpath.html

How classes are found:

http://download.oracle.com/javase/6/docs/technotes/tools/findingclasses.html

JDK Tools and Utilities:

http://download.oracle.com/javase/6/docs/technotes/tools/

Chapter 3

Wrappers

Each isCOBOL product, from Compiler to Application Server, is identified by a java class.

The standard way to launch these classes would be by adding the library in which they're stored to the Classpath and by issuing the command:

```
java ClassName
```

However, since this kind of approach would generate long commands and would require some efforts in the Classpath configuration, wrappers are provided along with isCOBOL.

These wrappers are identified by exe files on the Windows platform and by shell script files on other platforms.

They're stored in the isCOBOL bin directory.

When you launch a wrapper, the current directory (".") and all the jar libraries stored in the isCOBOL lib directory are automatically added to the Classpath, then your command is translated as follows:

```
wrapper_name Parameters
```

becomes

```
java iscobol_poduct_class Parameters
```

Standard wrappers

The following table lists all the standard wrappers, which are available for all platforms, followed by the corresponding isCOBOL class

Wrapper	Class
cobfileio	com.iscobol.lib.COBFILEIO
cpgen	com.iscobol.compiler.CopyGen
cpk	com.iscobol.lib.CPK
edbiis	com.iscobol.easydb.EdbiIs

Wrapper	Class
gife	com.iscobol.lib.GIFE
iscc	com.iscobol.compiler.Pcc
iscclient	com.iscobol.gui.client.Client
isconfig	com.iscobol.lib.ISCONFIG
iscbalancer	com.iscobol.balancer.LoadBalancer
iscremotecc	com.iscobol.compiler.remote.server.Server
iscrun	com.iscobol.invoke.Isrun
iscserver	com.iscobol.as.AppServerImpl
iscupdater	com.iscobol.updater.SoftwareUpdater
isl	com.iscobol.lib.ISL
ismigrate	com.iscobol.lib.ISMIGRATE
iscmigrate	com.iscobol.lib.ISMIGRATE
issort	com.iscobol.issort.IsSort
jdbc2fd	com.iscobol.lib.JDBC2FD
jutil	com.iscobol.lib.JUTIL
stream2wrk	com.iscobol.utility.Stream2Wrk

Based on the above tables, you see that, for example, using the command:

```
iscc
```

is the same as using:

```
java com.iscobol.compiler.Pcc
```

Windows wrappers

On Windows platform five additional wrappers are available.

Wrapper	Class
iscclientd	com.iscobol.clientlstnr.ClientListener
isclient	com.iscobol.gui.client.Client
isclientd	com.iscobol.clientlstnr.ClientListener

Wrapper	Class
isrun	com.iscobol.invoke.Isrun
isupdater	com.iscobol.updater.SoftwareUpdater

Note - isrun and isclient are the same as iscrun and iscclient except that they launch the isCOBOL class with javaw.exe instead of java.exe. In this way the java process does not keep the console busy.

With these wrappers that don't display output on the console, if an unexpected exception occurs or if the program displays data upon sysout and syserr, two files named *wrapper*_out.log and *wrapper*_err.log (where *wrapper* can be *isrun* or *isclient* depending on the exe you launched) are updated in the isCOBOL bin directory.

All Windows wrappers force the Windows look and feel for the application.

Launching:

```
isrun.exe PROGRAM
```

on Windows, is the same as launching:

```
java.exe -Dswing.defaultlaf=com.sun.java.swing.plaf.windows.WindowsLookAndFeel
   com.iscobol.invoke.Isrun PROGRAM
```

The -J option

The -J option allows you to pass options to the JVM instantiated by the wrapper. Each option specified by -J is placed between “java” and the class name in the command generated by the wrapper.

Example for Compiler

```
iscc -J-Discobol.compiler.const.const1=1 prog1.cbl
```

becomes:

```
java -Discobol.compiler.const.const1=1 com.iscobol.compiler.Pcc prog1.cbl
```

Example for Utilities

```
gife -J-Discobol.gife.efd_directory=/dev/myapp/efd
```

becomes

```
java -Discobol.gife.efd_directory=/dev/myapp/efd GIFE
```

Example for Runtime:

```
iscrun -J-Xmx512m -J-Discobol.conf=/myapp/conf PROG1
```

becomes:

```
java -Xmx512m -Discobol.conf=/myapp/conf com.iscobol.invoke.Isrun PROG1
```

Example for Debugger:

When you run the Debugger you must pay attention in using the -J option because there are two different virtual machines involved:

- the java virtual machine that runs the Debugger
- the java virtual machine instantiated by the Debugger, that runs the COBOL program

Options passed through -J are passed to the both java virtual machines, while options passed without -J are passed only to the second java virtual machine.

The following command, for example, shows how to pass the `iscobol.debug.code_prefix` property to the Debugger and a configuration file to the COBOL program:

```
iscrun -d -J-Discobol.debug.code_prefix=/myapp/src -Discobol.conf=/myapp/conf PROG1
```

That becomes:

```
java com.iscobol.invoke.Isrun -Discobol.debug.code_prefix=/myapp/src -d -
   Discobol.conf=/myapp/conf PROG1
```

The internal class com.iscobol.invoke.Isrun, if the -d option is used, behaves as follows: settings specified before the -d option are passed to the Debugger JVM and to the program JVM, instead settings specified after the -d option are passed only to the program JVM.

vmoptions files

An alternative way to pass Java options is by editing the wrapper's vmoptions file, where applicable.

The following table lists all the wrappers that are configurable through a vmoptions file along with the name of the corresponding vmoptions file.

The vmoptions file is placed in the same folder as the wrapper.

Wrapper	vmoptions file
iscrun	iscrun.vmoptions
isrun	isrun.vmoptions
iscclient	iscclient.vmoptions
isclient	isclient.vmoptions
iscserver	iscserver.vmoptions

Wrapper	vmoptions file
isserver	isserver.vmoptions
isbalancer	isbalancer.vmoptions
isremotecc	isremotecc.vmoptions

Options passed through -J are appended to the options found in the vmoptions file.

Chapter 4

Special Features

This chapter lists the most interesting special features that are offered by isCOBOL.

New syntax

The COBOL syntax supported by isCOBOL contains some interesting extensions introduced by the ANSI 2002 standard. With isCOBOL, programmers can take advantage of:

- Variable length alphanumeric items.

```
77 my-var PIC X ANY LENGTH.
```

- Dynamic occurs.

```
01 my-table OCCURS DYNAMIC CAPACITY num-items.
  03 my-item1 PIC X(10).
  03 my-item2 PIC 9(5).
```

- The ability to measure the length of a string into PIC X items with one single statement.

```
INSPECT my-var TALLYING var-length FOR CHARACTERS BEFORE INITIAL TRAILING SPACE.
```

- The ability to interact with Java objects and create new objects using the proper syntax.

```
SET the-result TO my-object:>my-method ( parameter-1, parameter-2 )
```

- The new statement ASSERT to raise exception if the assertion is false when executing with the -ea Java option. This is particularly useful for debugging purpose as in Visual C and Java languages. Code example:

```
ASSERT (var1 = 1 or var2 = 2)
     otherwise "Exception message to raise, " VAR1, VAR2.
```

I/O

isCOBOL offers some additional features for I/O on files and databases. With isCOBOL, programmers can take advantage of:

- Native support for embedded SQL syntax; no precompilers are needed.

```
EXEC SQL
   SELECT COUNT (*) INTO :rec-count FROM TABLE1
END-EXEC.
```

- National items are supported; data is stored using UTF-16 Big Endian encoding.

```
77 MY-VAR PIC N(10).
```

- The ability to interact with databases while maintaining standard COBOL statements (see for details).
- The ability to associate sequential files with the standard input, output and error.

```
SELECT stdin ASSIGN TO "-S IN"
    ORGANIZATION LINE SEQUENTIAL.

SELECT stdout ASSIGN TO "-S OUT"
    ORGANIZATION LINE SEQUENTIAL.

SELECT stderr ASSIGN TO "-S ERR"
    ORGANIZATION LINE SEQUENTIAL.
```

- The ability to create PDFs or previews of print files.

```
SELECT pdf-file ASSIGN TO PRINT "-P PDF /usr/docs/print.pdf"
    ORGANIZATION LINE SEQUENTIAL.

SELECT ptr-prev ASSIGN TO PRINT "-P PREVIEW"
    ORGANIZATION LINE SEQUENTIAL.
```

Routines and functions

Below is a list of special features offered by isCOBOL subroutines and functions:

- The ability to create icons for menu items (see WMENU-ADD-BITMAP, WMENU-CHANGE-BITMAP and WMENU-DELETE-BITMAP for details).
- The ability to capture the current screen (see W$CAPTURE for details).
- The ability to scale and rotate pictures and inquire their size (see W$SCALE, W$ROTATE and W$IMAGESIZE for details).
- The ability to load fonts directly from ttf files without installing them (see W$CREATEFONT for details).
- The ability to know which resource is associated with a handle (see FUNCTION HANDLE-TYPE for details).
- The ability to retrieve current ip and machine names (see J$NETADDRESS for details).
- The ability to backup and restore the current environment situation (see C$ENVMAP for details).
- The ability to retrieve the list of all configuration properties currently set (see C$LIST_ENVIRONMENT for details).
- The ability to remove a property from the configuration at run time (see C$UNSET for details).

Distributed environment (Application Server)

isCOBOL provides the ability to deploy the COBOL application in an Application Server environment. This kind of approach separates the backend part (that will run server-side) from the UI part (that will run client-side).

This kind of achitecture runs on every kind of network (local and remote) using the TCP/IP protocol.

While working in an Application Server environment, the program can take advantage of the following features:

- The ability to execute programs client side.

```
CALL CLIENT "MYPROG" USING param-1, param-2.
```

- The ability to run programs on remote machines (see Remote objects for details).
- The ability to read and write binary files on the client machines; files must be defined as follows:

```
SELECT client-file ASSIGN TO filename
  ORGANIZATION BINARY SEQUENTIAL
  CLASS "com.iscobol.io.RemoteRelative"
  .
```

- The ability to copy each kind of file (sequential, binary and indexed) from client to server and viceversa (see C$COPY for details)

GUI

isCOBOL offers some additional features that allows you to produce complex and flexible graphical user interfaces. The programmer can take advantage of:

- The ability to render html in most all controls. The following code snippet shows how to show an animated GIF on the screen using the LABEL control:

```
03 LABEL
  line   2
  col   25
  lines  5 cells
  size   9 cells
  title  '<html><img src="file:Files/lb.gif"></img></html>'
  .
```

- The ability to create tooltips for controls, with the new HINT property.

```
03 ENTRY-FIELD
   line     2
   col      25
   size     9 cells
   value    w-name
   hint     "Write the name here"
   .
```

- The ability to use RGB values to set colors; e.g.:

```
03 LABEL
   line              2
   col               25
   size              9 cells
   title             "gray label"
   background-color  rgb x#c0c0c0
   .
```

- The ability to load GIF, PNG as well of BMP and JPG pictures for use both on the screen and while printing.
- The ability to retrieve user selected text from an entry-field. The following code snippet retrieves the text selected by the user from the entry-field named ef-1:

```
INQUIRE ef-1 SELECTION-TEXT w-text.
```

- The ability to create a mask to help the user inputing data.

```
03 ENTRY-FIELD
   line     2
   col      25
   size     9 cells
   value    w-date
   format-string "##/##/####"
   .
```

- The ability to display vertical labels.

```
03 LABEL
   vertical
   line      2
   col       25
   lines     15 cells
   size      8 cells
   title     "vertical text"
   .
```

- The ability to reorder and sort GRID columns (see Reordering-Columns and Sortable-Columns for details)
- The ability to copy the GRID content to clipboard and export it to xls/xlsx spreadsheets. See Heading-Menu-Popup.
- The ability to select multiple rows and columns in a GRID. See Selection-Mode.
- The ability to display more lines of text in a single GRID cell. Alignment of the column with multiline text must be “H”.
- The ability to merge cells in the GRID header. See Cell-Columns-Span and Cell-Rows-Span.
- The ability to mix different DATA-TYPES in the same column, for example:

```
DATA-TYPES ("U(1)L(0)", "9(3)X(2)")
```

- The ability to display controls inside GRID cells as explained at GRID.
- The ability to protect GRID cells both with read-only and skip approaches (see Row-Protection, Column-Protection and Cell-Protection for details)
- The ability to show and hide GRID rows and columns dynamically (see Row-Hiding and Column-Hiding for details)
- TREE-VIEW items can be edited by the user.
- The ability to add icons to COMBO-BOX items.

```
MODIFY ComboBoxHandle, ITEM = 1, BITMAP-NUMBER = 20
```

- The ability to add icons to TAB-CONTROL page labels.

```
MODIFY TabControlHandle, TAB-INDEX = 1 BITMAP-NUMBER = 1
```

- The ability to dynamically add and remove pages on TAB-CONTROL (see Insertion-Index and Tab-To-Delete for details)
- The ability to change the text of TAB-CONTROL page labels without re-creating the page (see Tab-Text for details)
- The ability to disable a TAB-CONTROL page to prevent users to select it (see Tab-Enabled for details)
- A new style for TAB-CONTROL, Allow-Container, that allows to bind screen entries to TAB-CONTROL pages and have the page switch managed automatically by the runtime.

- A brand new graphical control: the SLIDER.
- The ability to intercept mouse events on BITMAP control (see MSG-MOUSE-CLICKED, MSG-MOUSE-ENTER and MSG-MOUSE-EXIT for details).
- The ability to interface JAVA-BEAN controls.
- The ability to show a custom icon on graphical windows (see Icon for details).
- The ability to add an icon to STATUS-BAR panels and to color them with different colors(see Panel-Bitmap, Panel-Bitmap-Number, Panel-Bitmap-Width, Panel-Bitmap-Alignment and Panel-Color). It's also possible to align the panel text inside the panel (see Panel-Alignment).
- The ability to show both text and icon on a PUSH-BUTTON (see Title-Position for details).
- The ability to unplug the TOOL-BAR from the window with the MOVEABLE style.

```
DISPLAY TOOL-BAR MOVEABLE
        HANDLE toolbar-handle.
```

- The ability to create docking windows. This means that multiple windows can be attached to a single container window and they will resize accordingly.

  The syntax to create the container is:

```
display docking window
                layout w-layout
                [...]
                handle h-main.
```

  The syntax to create a window inside the container is:

```
display dockable window
                 upon h-main
                 upon-leaf leaf-name
                 [...]
```

- The ability to create MDI windows. This means that multiple windows can be included into a single container window.

  The syntax to create the container is:

```
display mdi-parent window
                [...]
                handle h-main.
```

The syntax to create a window inside the container is:

```
display mdi-child window
                 upon h-main
                 [...]
                handle h-child.
```

- The ability to pop up a list of possible values while the user is editing an EntryField (see Proposal).
- The ability to display bitmaps inside an EntryField and to have events upon mouse over and mouse click on these bitmaps.
- The ability to show a placeholder text within EntryField and ComboBox.
- The ability to copy text from character based screens and to paste text to character based Accept.
  The user can select text from the screen by dragging the mouse with left button hold. The text is automatically copied in the clipboard as soon as the user releases the mouse button. The user can also paste some text from the clipboard by pressing the middle mouse button (usually identified by the scroll wheel); the pasted text is put in the keyboard buffer and the active ACCEPT gets it.
- The ability to have more row headings in the Grid with the property Num-Row-Headings.
- The ability to intercept new events MSG-ICONIFIED and MSG-DEICONIFIED for the Window when user reduces the window to task bar or restores it.
- The ability to automatically scale a picture in the Bitmap control with the property Bitmap-Scale.
- An easier management of Tab-Control pages through the style Allow-Container.
- The ability to show a Tab-Control as an Accordion container.
- The ability to create modern tool-bars, also known as RIBBON.
- The ability to color the background of a window with a gradient effect:

```
     *a window whose background color goes from gray to white
     *the code mixes the use of rbg color values and COBOL color values
          display standard graphical window
                  gradient-color-1 rgb x#c0c0c0
                  gradient-color-2 16
                  gradient-orientation 0
```

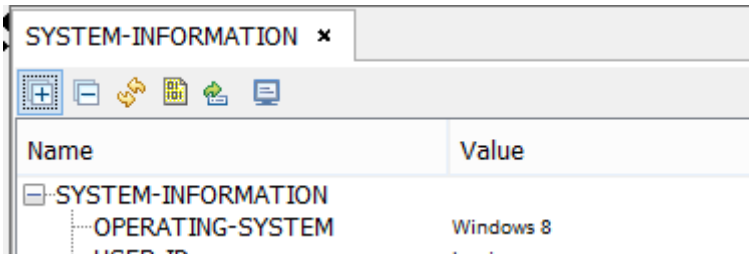
For a quick demonstration of most GUI fSeatures, launch the isCOBOL Demo program.

Debugger

- The isCOBOL Debugger shows the source code using different colors for keywords, strings and literals.
- The isCOBOL Debugger shows the content of copybooks with a different background color, to easily distinguish them from the rest of the source code. When the COPY statement contains the REPLACING clause, the isCOBOL Debugger shows the result of the REPLACING instead of the file content.

```
 4 program-id.                 isControlSet.
 5
 6 configuration section.
 7 copy "copylib/classes.rep".
 1*> Copyright (c) 2010 by Veryant LLC. Users of isCOBOL APS
 2*> may freely modify and redistribute this program.
 3
 4 repository.
 5   class BorderLayout        as "java.awt.BorderLayout"
 6   class JavaBean            as "com.iscobol.gui.server.CobolGUIJavaBean"
 7   class ChartFactory        as "org.jfree.chart.ChartFactory"
 8   class ChartPanel          as "org.jfree.chart.ChartPanel"
 9   class JFreeChart          as "org.jfree.chart.JFreeChart"
10   class PiePlot3D           as "org.jfree.chart.plot.PiePlot3D"
11   class XYSeries            as "org.jfree.data.xy.XYSeries"
12   class XYSeriesCollection  as "org.jfree.data.xy.XYSeriesCollection"
13   class DPDataset           as "org.jfree.data.general.DefaultPieDataset"
14   class DCDataset           as "org.jfree.data.category.DefaultCategoryDataset"
15   class PlotOrientation     as "org.jfree.chart.plot.PlotOrientation"
16   .
 8 special-names.
 9   decimal-point is comma.
10
11 input-output section.
12 file-control.
13     select data-gui assign to ".infogui" status file-status.
14
15 file section.
16 fd  data-gui.
17 01  rec-data-gui.
18     03 bk-page              pic 9.
19     03 bk-displ             pic x(40).
```

- The isCOBOL Debugger allows you to set and inquire graphical controls properties as well as DATA DIVISION variables.
- The isCOBOL Debugger can display group variables as a tree, to easily monitor the content of each item of the group.

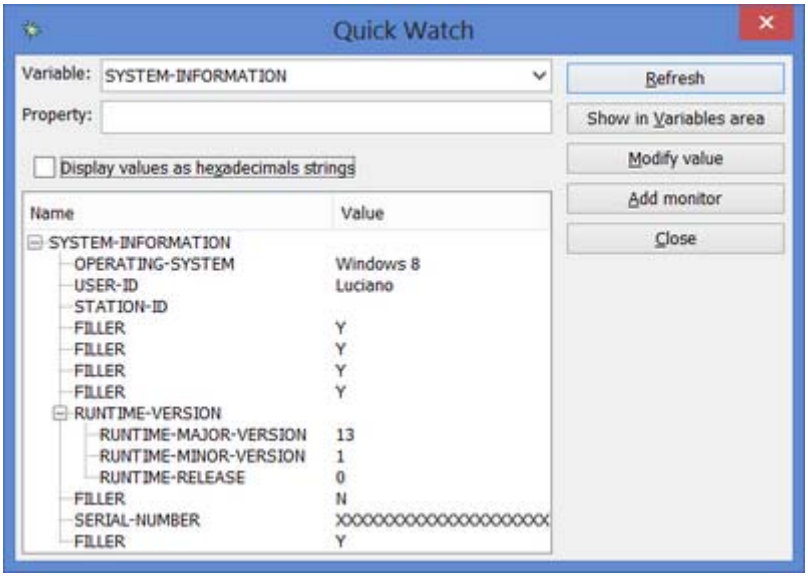


- The isCOBOL Debugger allows you to select one or more lines of source code and copy them into the

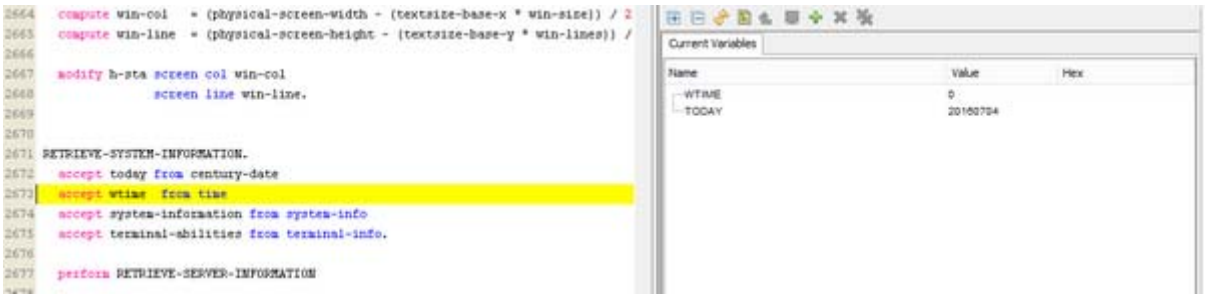
clipboard.

```
96      into buffer
97    end-string
98
99    display standard graphical window
100           title "isCOBOL ControlSet"
101           lines 36,6
102           size 117
103           min-lines 36,45
104           min-size 117
105           screen line win-line
106           screen col win-col
107           control font h-font
108           background-low
109           handle h-sta
110           visible 0
111           resizable
```

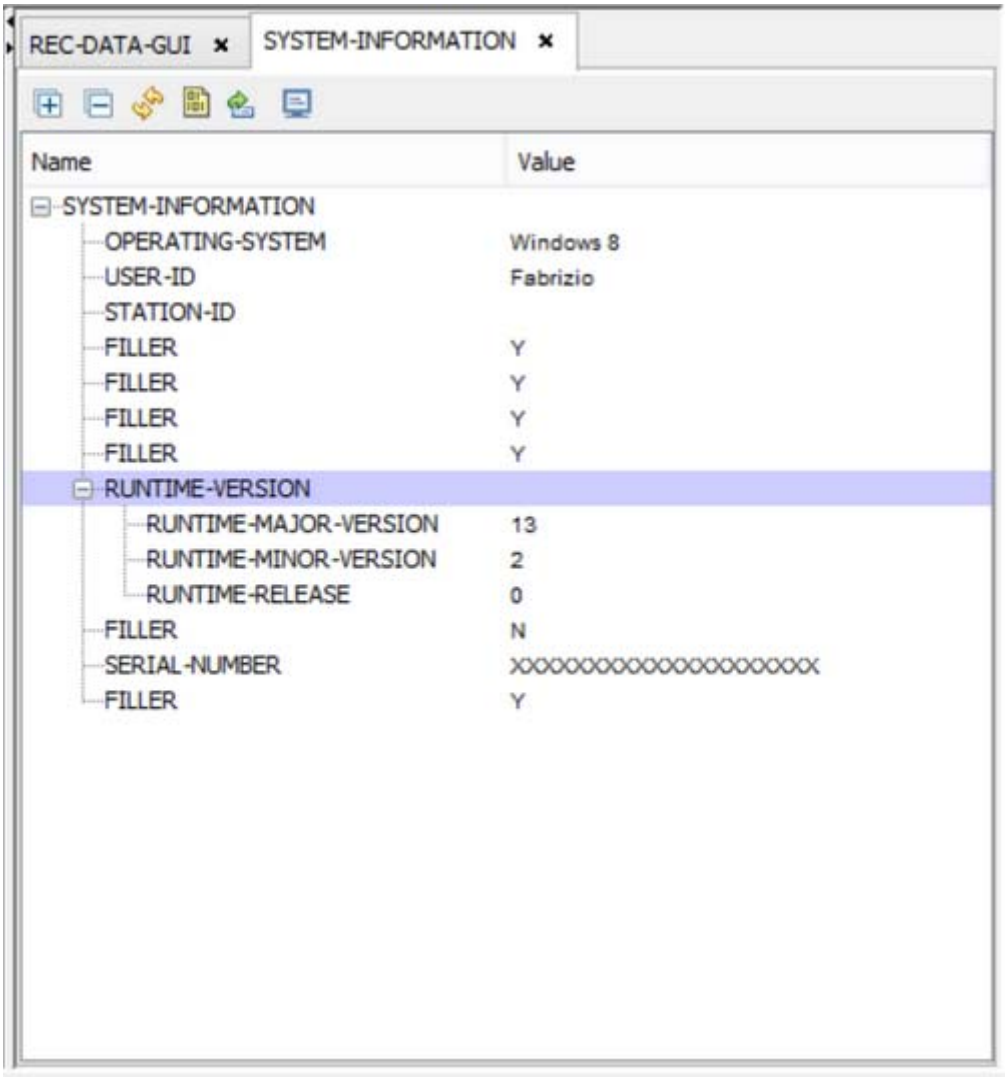

- A quick watch feature is provided to easily handle DATA DIVISION variables and control properties.



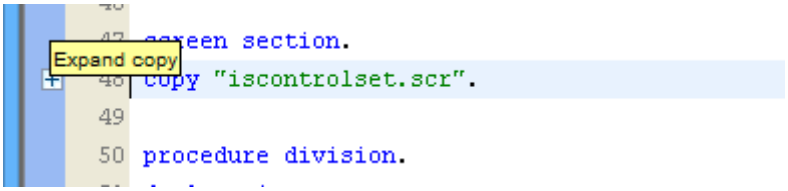
- A current variables area constantly updated as you step through statements, so you can monitor the value of the variables involved in the debugged statements:



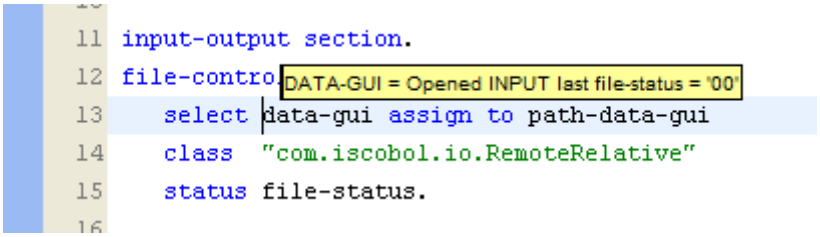
- A variable area that allows to monitor multiple data items at once:



- The ability to expand and collapse copy files:



- The ability to check files state during the debug session:



- The isCOBOL Debugger can debug programs on the local pc as well as programs on remote machines.

For detailed information about the above features, consult the Debugger section in User's Guide.