



isCOBOL™ Application Platform Suite

Frequently Asked Questions

Copyright © 2008 Veryant.
2415 East Camelback Road, Suite 700, Phoenix, AZ 85016, U.S.A.

All rights reserved.

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

isCOBOL is a trademark or registered trademark of Veryant in the U.S. and other countries.

Table of Contents

Executive Overview	5
1. What is the isCOBOL Application Platform Suite?.....	5
2. What are the benefits of an isCOBOL APS approach?.....	5
3. How compatible is isCOBOL APS with my current COBOL?	5
4. What platforms does isCOBOL APS support?	6
5. My company is concerned about the future of COBOL, how does isCOBOL APS alleviate that concern?.....	6
Development Environment	6
6. What impact does isCOBOL APS have on developers?	6
7. What options are there for debugging isCOBOL environments?.....	8
8. Is there an IDE available with isCOBOL APS?.....	8
9. What data access options exist with isCOBOL APS?.....	9
10. Can data sources be mixed and matched within an application?	11
11. What advantages are there to using the Embedded SQL compiler available with the isCOBOL environment?	11
12. Suppose I find a data inefficiency and want to write a new version of 'get data,' is that possible?.....	11
Moving to isCOBOL APS	11
13. How hard is it to move to isCOBOL APS?	11
14. My code is sprinkled throughout the programs for Thin Client. What might I expect for a migration path for isCOBOL APS in this area?	12
15. What user interface (UI) features are not supported by isCOBOL APS?	12
isCOBOL APS and Java Technology.....	13
16. How does isCOBOL APS run COBOL programs in a Java environment?	13
17. After moving to isCOBOL APS, what am I maintaining – COBOL or Java source code? Also, what happens to highly commented COBOL code?.....	13
18. I've heard of COBOL being wrapped or converted to Java before, but developers tell me it is just turned into spaghetti code. What is unique about isCOBOL APS?.....	15
19. What about performance in general? Are there any performance issues moving from COBOL to Java technology?	15

20. I've seen specific issues where Embedded SQL (ESQL) generated at the Java level needs to be tweaked in order to perform correctly. How does isCOBOL APS address this? 16

21. How does isCOBOL APS facilitate integration between COBOL and non-COBOL application environments and platforms? 16

isCOBOL Thin Client 17

22. What if I want to distribute my isCOBOL application out to several different devices with a variety of look and feel, connectivity and file system requirements? 17

23. Can I run standard syntax COBOL programs at the PDA/CDC level with isCOBOL APS? 17

Executive Overview

1. What is the isCOBOL Application Platform Suite?

Supporting the latest ANSI and legacy COBOL dialects, the isCOBOL Application Platform Suite (isCOBOL APS) is a complete environment for COBOL application development and deployment. In addition to a robust compiler, integrated development environment (IDE), debugging facility and numerous data access options, isCOBOL APS has a portable graphical user interface (GUI) and is able to execute on any device that supports a Java Runtime Environment (JRE) version 1.4.2 or greater – from mainframes to mobile phones. By compiling COBOL code into Java class output, isCOBOL APS allows you to retain and enhance valuable COBOL application and development assets while taking full advantage of Java technology in deployment.

2. What are the benefits of an isCOBOL APS approach?

isCOBOL APS maintains current COBOL investments while simultaneously providing a path forward to take full advantage of the openness, portability and power of Java technology without retraining or rewriting. With a complete suite of offerings and an object-orientated development approach, isCOBOL technology delivers cost-effective, scalable environments that can be easily integrated and extended to meet the demands of today's rapid development, 'always on' world. With isCOBOL APS, you no longer need worry about having to replace COBOL 'someday,' instead isCOBOL APS provides the flexibility to choose the best programming language for your particular job at hand.

3. How compatible is isCOBOL APS with my current COBOL?

isCOBOL APS was developed by a talented team of COBOL and Java experts with extensive knowledge of COBOL dialects. isCOBOL APS is designed to support your current COBOL language syntax and features and is highly compatible with today's common COBOL dialects.

As a service to you, Veryant can take a sample of your code and generate a Code Analysis Report that will tell you exactly how compatible your application is with isCOBOL APS before any investment occurs on your side.

4. What platforms does isCOBOL APS support?

Being Java-based, the isCOBOL™ Compiler runs on any platform that supports a Java Development Kit (JDK) version 1.4.2 or greater. The isCOBOL Debugger and isCOBOL compiled code can run on any platform that supports a Java Runtime Environment (JRE) of 1.4.2 or later. Thus, isCOBOL APS supports a wealth of platforms including AIX, HP-UX, Linux, Solaris and Windows. The only exception to this is isCOBOL™ ISAM ODBC, which currently runs only on Windows.

5. My company is concerned about the future of COBOL, how does isCOBOL APS alleviate that concern?

Besides the natural benefit of serving as another choice in the COBOL market, isCOBOL APS preserves valuable COBOL code while offering companies the advantage of the Java platform. With the ability to treat COBOL programs as Java classes, your COBOL programs can be utilized by other Java classes just as if they were originally native Java code. Until then, Java can serve as the platform while COBOL serves as the language. The only items required to execute the outputted Java code are the JVM and the iscobol.jar file that implements the original COBOL verbs.

The Java community is vibrant and diverse, and the isCOBOL Compiler is built on top of a Java compiler that is over 12 years old. By 2005, it was estimated that approximately 4.5 million developers used Java technology and that over 2.5 billion Java technology-enabled devices were available worldwide⁽¹⁾. Java technology has recently been made freely available under the GPL license and that has only further opened new development opportunities for software developers. These figures, combined with the broad reaching industry support for Java from corporations such as IBM, Sun, HP, BEA, Oracle, Google and others ensures the security of Java technology for decades to come.

⁽¹⁾*The Java History Timeline - <http://www.java.com/en/javahistory/timeline.jsp>*

Development Environment

6. What impact does isCOBOL APS have on developers?

isCOBOL APS preserves COBOL assets and does not require staff to be retrained in a different programming language. COBOL developers continue to use the IDE or editor of their choice, as well as desired command line utilities and scripts for source code control and builds, thus isCOBOL technology should have no impact on development.

isCOBOL developers leverage current COBOL skills and can deploy applications on the Java platform without needing to know how to program in the Java language. isCOBOL programs are indistinguishable from programs written in the Java language because the isCOBOL Compiler converts COBOL source code to Java source code behind-the-scenes. If it makes business sense, developers can transition from COBOL programming to Java programming over time. isCOBOL APS gives you the ability to keep your COBOL assets even if you choose Java as your programming language at some time in the future.

For deployment, a few commonly known Java platform items primarily around configuration, such as CLASSPATH, JDK, JRE and the Java executable names will need to be known:

- CLASSPATH is an environmental variable which tells the JVM where to find the class libraries, including user-defined class libraries.
- The Java Development Kit (JDK) is a software development environment for writing applets and applications in the Java programming language and contains the Java compiler.
- The Java Runtime Environment (JRE) is a subset of the JDK. It is for end-users and developers who want to redistribute the runtime environment alone. The Java runtime environment consists of the Java virtual machine, the Java core classes and supporting files and has everything needed to run Java programs.

If you choose to further extend isCOBOL applications with Java technology, the path is made easier with isCOBOL APS. Since isCOBOL APS preserves current COBOL programs, the transition to Java can be done at your desired pace.

Given the numerous tools and options available in Java, developer productivity may be enhanced by moving to isCOBOL APS as well. For instance, the Java platform offers a range of options when it comes to areas such as logging, tracking, debugging, monitoring statistics and resource allocation, as well as profiling to help identify areas for application performance tuning.

Note: the terms above, and other common Java terms are defined online at <http://Java.sun.com/docs/books/tutorial/information/glossary.html>

7. What options are there for debugging isCOBOL environments?

The isCOBOL™ Runtime Environment includes a robust debugging tool for continued COBOL development. Written in Java, the isCOBOL™ Debugger provides graphical, COBOL source level debugging and runs on any platform on which the Java Runtime Environment (JRE) 1.4.2 or later is available. Since the isCOBOL Debugger is included with the isCOBOL runtime library, you do not need to purchase a different compiler for every operating system you wish to deploy on.

From the isCOBOL Debugger, you can perform activities such as trace-file writing, loading and unloading sessions, executing instructions, searching for text and setting breakpoints.

The debugger window provides three primary code views:

- The *source code window* enables source code viewing.
- The *command window* allows developers to view all executed commands and messages, as well as insert commands.
- The *info window* displays the execution stack, variables to monitor and the breakpoint list.

isCOBOL Debugger windows can be resized as desired and debugger commands can be issued using the command-line, menu items or via toolbar buttons.

Remote debugging can also be used to debug programs that are executed on a server. In this case, programs would need to be compiled with a '-d' option, then accessed from a client machine that has the appropriate source files running on it.

8. Is there an IDE available with isCOBOL APS?

With isCOBOL APS, developers can continue to use the editor of their choice, command line utilities and scripts for source code control and builds. If you are looking for an IDE to provide a powerful and adaptable graphical interface, the isCOBOL IDE is also available. The isCOBOL IDE is Eclipse-based and offers a single, flexible environment for all development tasks (design, coding, testing, debugging). You can move, dock, undock and hide interface elements as well as customize keyboard shortcuts as desired in the isCOBOL IDE.

The isCOBOL IDE also includes a dynamic syntax checker that checks program syntax while a developer is coding and provides real-time alerts to potential code violations

before compilation has occurred. This real-time checker, in addition to the code hints that the isCOBOL IDE provides, enhances productivity and greatly reduces edit, compile, and correction cycles.

9. What data access options exist with isCOBOL APS?

There are multiple ISAM file system and database options available with isCOBOL APS. The isCOBOL environment offers isCOBOL™ ISAM Server if you are looking to deploy in a robust client/server environment. If you are looking to deploy in a compact, portable environment, the 100% Java-based isCOBOL™ JISAM (a Java-based index file system) is an additional file system option for isCOBOL APS.

In terms of RDBMS options, isCOBOL APS supports major databases such as Oracle, DB2, MySQL and Microsoft SQL Server. The isCOBOL Compiler automatically translates ESQL to JDBC, and the isCOBOL™ ESQL Generator is available if you wish to move from ISAM files to a relational database without changing COBOL code or learning ESQL.

The following list briefly describes each file system option:

isCOBOL™ ISAM Server

- High capacity, scalable, transactional, client/server indexed file data access.
- Written in C/C++, available on Windows, aix4, aix5-32, aix5-64, hpux, linux, macosx, sco, soli386, solsparc-32
- Full transaction support
- Includes ODBC driver for Windows
- Supports Live Backup
- Includes several feature-rich management utilities such as ISMIGRATE.
- Technical characteristics:
 - Maximum file size: 16,700,000 terabytes (16 Exabytes)
 - Maximum number of keys: no limit
 - Maximum number of records: no limit
 - Maximum key length: no limit
 - Maximum number of segments per key: 16
 - Maximum fixed record length: 65,535 bytes
 - Maximum variable record length: 16,700,000 terabytes (16 Exabytes)

isCOBOL™ JISAM

- Supplied with isCOBOL APS

- Written entirely in Java, so it runs anywhere, even on a mobile phone
- Utility ('JUTIL') provided for basic file management
- Utility ('ISMIGRATE') provided for one-step migration of data files from Vision
- Technical characteristics:
 - Maximum file size: 2GB
 - Maximum number of keys: no limit
 - Maximum number of records: no limit
 - Maximum key length: 256 bytes
 - Maximum number of segments per key: 8
 - Maximum record length: 32,767 bytes

Other RDBMS

- Other supported RDBMS options include any that provide a JDBC driver. Oracle, DB2, MySQL, Microsoft SQL Server and DBMaker have been tested and verified with isCOBOL APS. These databases can be accessed with COBOL I/O statements or ESQL using the ESQL to JDBC automatic translation available in the isCOBOL Compiler and/or the isCOBOL ESQL Generator available with the isCOBOL APS environment.
- The isCOBOL Compiler recognizes ESQL and translates it into Java source code, which then directly interfaces with JDBC., so no database-specific pre-compilation is required. Advantages to this approach are that you now have open access to the COBOL/ESQL source code which becomes part of your application. The interface to the database (i.e. the JDBC driver) is provided by and supported by the database vendor, and there is no need to relink runtime libraries.
- The isCOBOL™ ESQL Generator is a 4GL technology that reads a specified COBOL file description from existing COBOL source files and generates an isCOBOL file system interface implemented with ESQL in COBOL. When you run a program, the COBOL file I/O operations are routed through the generated COBOL/ESQL interface with talks to the RDBMS through the JDBC driver provided by the database vendor. There is no need to change your COBOL code or learn ESQL. This method often proves preferable to middleware interface layers because now you have open access to the COBOL/ESQL source code which becomes part of your application. This is comparable to other 4GLs which gain access through a proprietary C interface layer.

10. Can data sources be mixed and matched within an application?

Yes. In isCOBOL APS you can select the data source per file as well as globally by adding an entry in the runtime properties file.

11. What advantages are there to using the Embedded SQL compiler available with the isCOBOL environment?

With the isCOBOL Compiler, developers only need to get the JDBC driver for a specific database, specify this in the config file, and then the isCOBOL program will make calls to this JDBC driver. JDBC drivers provide a robust and mature method of accessing popular relational databases such as Oracle and DB2.

12. Suppose I find a data inefficiency and want to write a new version of 'get data,' is that possible?

Yes, with isCOBOL APS, you have the ability to intercept COBOL I/O operations to extend or replace their implementation with your own. The first step we would take here would be getting to know your specific situation.

Moving to isCOBOL APS

13. How hard is it to move to isCOBOL APS?

When considering a move to isCOBOL, Veryant can take a sample of your code and generate a complimentary Code Analysis Report that will tell you exactly how compatible your application is before any investment occurs on your side.

In terms of data, if data resides in flat files or is supported by a database such as Oracle, there is no data migration involved. If data resides in an indexed file system it can be moved to isCOBOL JISAM, isCOBOL ISAM Server or another database. To migrate data to another file system or database, you write a simple isCOBOL program that reads all of the records from your current file system and writes them to the file system or database of your choice, or use the included isCOBOL APS 'ISMIGRATE' utility which does this for you.

If you want to access an RDBMS such as Oracle or MySQL you can use the isCOBOL ESQLE Generator to generate an isCOBOL file system interface implemented with embedded SQL in COBOL. To do this, you add "iscobol.file.index=easydb" and the JDBC related properties for the particular database you want to use to your *iscobol.properties* file.

When you run, COBOL I/O operations will be routed through the isCOBOL ESQL Generator-produced interface which talks to the RDBMS through the JDBC driver provided by the database vendor.

Here are a few of the considerations that will need to be kept in mind during a move to isCOBOL APS:

- If COBOL source code is compiled with a batch file, shell script or makefile, the compiler name will need to be adjusted to *iscc*.
- Some compiler options may need to be adapted or renamed to be supported in the isCOBOL environment.
- Numeric format will need to be declared:
 - `-dci` IBM numeric format
 - `-dcd` Data General binary numeric format
 - `-dca` ACUCOBOL-GT numeric format
 - `-dcm` Micro Focus numeric format

Transition services are also available for companies who would like to supplement in-house staff during such a move. For additional details, contact info@veryant.com.

14. My code is sprinkled throughout the programs for Thin Client. What might I expect for a migration path for isCOBOL APS in this area?

With the isCOBOL environment, user interface developers would never face a case where they would have to write a user interface one way for Windows, another way for UNIX and Linux, etc. Since isCOBOL APS is Java based, developers get a deployable GUI, regardless if the client is Windows, UNIX, Mac, handheld, etc. You can select to run programs locally or with isCOBOL™ Server, which provides Thin Client and Distributed functionality. Either way, you can use the identical user interface. With isCOBOL APS, programs running on the server can call programs on the client with the CALL CLIENT statement.

15. What user interface (UI) features are not supported by isCOBOL APS?

The isCOBOL Compiler supports the oldest and the newest COBOL syntax for user interfaces, including ANSI and later variations of text-mode ACCEPT and DISPLAY, screen sections with FROM, TO and USING fields, and a full range of graphical windows

and controls. There is support for user interfaces originally written for dumb terminals, terminal emulators, and character-based consoles, as well as for graphical systems. The isCOBOL application user interface (UI) is fully portable, whether graphical or text-mode. The UI can run on any front-end machine that supports Java or X Windows. This includes Windows PCs, Macs, Linux/UNIX desktops, diskless and data less thin clients, X terminals and other virtual display clients such as Sun Ray, handheld devices, and more.

ActiveX is not supported the same way with the isCOBOL Compiler as with other COBOL offerings. isCOBOL APS supports object-orientated COBOL syntax, and when you are talking to objects such as with COM, OLE and ActiveX, it may be more natural to use object syntax. The isCOBOL Compiler produces pure Java code. To use COM, OLE or ActiveX objects with a Java-based program you need to use a third party product such as ComfyJ from TeamDev, Ltd. to generate wrapper classes to expose those objects to Java.

Support for an actual hardware dumb terminal will depend on the specifics of your application.

isCOBOL APS and Java Technology

16. How does isCOBOL APS run COBOL programs in a Java environment?

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 APS. The isCOBOL Compiler translates COBOL source code into Java source code every time you compile. The isCOBOL Compiler then feeds this Java source code into the Java compiler which produces Java bytecode objects which can be executed with a JVM. This process is transparent for developers. Although the Java source code is temporary intermediate output, you can compile it with a special “-jj” option if you ever want to retain this code.

17. After moving to isCOBOL APS, what am I maintaining – COBOL or Java source code? Also, what happens to highly commented COBOL code?

With isCOBOL APS, it is up to you as to whether or not the best route forward is to continue programming and maintaining code in COBOL or the Java programming language. isCOBOL APS enables you to choose between continuing to use proven

COBOL development code for as long as appropriate, or moving to Java technology when that becomes a better fit for your business.

In terms of highly commented COBOL code, if you rely on commented areas, need to fix bugs, or add new functionality, you can continue to perform these operations in COBOL for as long as desired. There is no need to read or maintain Java intermediate source because you can extend isCOBOL through Java technology without it. By default, the Java intermediate source files are temporary and are deleted by the isCOBOL Compiler when it is finished using them. However, you can specify the “-jj” compiler option at any time if you want to save these Java source files.

If you want to switch from COBOL programming to Java programming down the road, your business would likely wrap its COBOL programs in objects using Object-Oriented COBOL or Java for future use. In this case, your Java programmers would write new code in the Java programming language that would access the Java objects produced from the isCOBOL code. After compilation with isCOBOL Compiler, the COBOL programs are Java classes (i.e. .class files), and can be used by other Java classes without regard for the fact that they were originally written in COBOL. This way a Java developer would never need to look at the source because they would treat the COBOL at that point just like any other Java object.

Here is an example of a COBOL statement and the corresponding Java intermediate source code:

COBOL:

```
77 MY-DATA-ITEM PIC X(100).  
...  
INSPECT MY-DATA-ITEM REPLACING TRAILING SPACES BY LOW-VALUE.
```

Java intermediate code:

```
private com.iscobol.types.PicX MY-DATA-ITEM;  
...  
MY_DATA_ITEM=Factory.getVarAlphanum(wsMem,0,100,false,null,  
null,null,"MY-DATA-ITEM",false,false);  
...  
MY_DATA_ITEM.inspect($485$ , $486$);  
...
```

```
static final    PicX $485$=Factory.getStrLiteral(" ");
static final    LiteralAll
                $486$=Factory.getAllLiteral(new byte[] {0});
```

Notice that MY-DATA-ITEM has been converted to an object, and the INSPECT statement has been converted to code which invokes the 'inspect' method of the MY-DATA-ITEM object. The 'inspect' method takes two parameters which are generated as Java objects later on in the source.

18. I've heard of COBOL being wrapped or converted to Java before, but developers tell me it is just turned into spaghetti code. What is unique about isCOBOL APS?

isCOBOL APS is not an 'either or' proposition. Developers are not forced to choose between programming either in COBOL or in Java. The isCOBOL Compiler creates pure Java immediate source code and then calls the Java compiler which translates the code into pure Java bytecode as the output. The clean, efficient Java code output is still COBOL-like in nature and can be called as Java classes.

Going forward, most organizations would likely wrap their COBOL programs as objects. Thus, you would not need to look at the source in the long-term, but could instead create object-oriented interfaces to the code. With isCOBOL APS, the wrapping can be done in Object-Oriented COBOL or Java with the same results. The resulting objects are stored as bytecode in Java class files, and one cannot distinguish between objects wrapped one way or the other.

19. What about performance in general? Are there any performance issues moving from COBOL to Java technology?

The world's foremost technology-driven companies have selected Java as their platform of choice for performance-critical web-serving, application-serving, application-integrating and development products such as BEA WebLogic, IBM WebSphere, Oracle Application Server and more.

isCOBOL APS and the Java platform were designed from the ground up to support concurrent, multithreaded processing. With basic concurrency support in the Java programming language and in Java class libraries, the isCOBOL runtime library will take advantage of today's multithreaded, multi-core systems.

The speed of the Java Virtual Machine has improved with every release and it is now highly tuned and optimized. With the latest releases of the JVM, performance of Java matches, and in many cases surpasses, the speed of natively compiled languages.

In its early days, Java was criticized because programs ran slower and required more memory than those written in natively compiled languages such as C or C++. That criticism spurred JVM companies to devote a disproportionately large amount of resources towards improving Java performance, leading to dramatic improvements in the performance of Java programs over time.

The introduction of an adaptive optimization Just-in-time (JIT) compiler into the JVM alone accounted for a 10-fold increase in execution speed. And JVMs nowadays are “ergonomic” in the sense that they automatically self-tune for best performance based on the platform configuration.

When additional tuning is needed, the Java platform provides more tools and tuning options than perhaps any other language or technology to help you achieve the best possible performance.

You can view additional information about Java technology and Java performance tuning at <http://Java.sun.com> and other sources on the Web.

20. I've seen specific issues where Embedded SQL (ESQL) generated at the Java level needs to be tweaked in order to perform correctly. How does isCOBOL APS address this?

You would go to the COBOL source. With the isCOBOL ESQL Generator, I/O statements are translated into COBOL source code that includes ESQL, which the compiler then translates into JDBC calls. The ESQL resides in the isCOBOL ESQL Generator produced file which becomes part of your application source code.

21. How does isCOBOL APS facilitate integration between COBOL and non-COBOL application environments and platforms?

Executed production files in an isCOBOL APS environment are 100% Java-based, thus they inherit all the interoperability and network benefits commonly associated with Java technology. One of the key value propositions of the Java platform is “Compile once, run anywhere.” Java support is now found in all major operating systems, and it is built into popular web browsers and consumer devices.

Java has been designed from the ground up to be network-centric. Its class structure supports the development of web services and provides a foundation for enterprise

development. The extensive set of standard application programming interfaces (APIs) supplied with the Java platform support almost everything you might want your applications to do -- from CORBA connectivity to internationalization.

isCOBOL Thin Client

22. What if I want to distribute my isCOBOL application out to several different devices with a variety of look and feel, connectivity and file system requirements?

With isCOBOL APS, it is relatively simple to distribute the same program with different look and feels; a different set of graphical control behaviors; a different set of error codes; different file systems, etc. This is made possible by adjusting settings in the framework properties file and by taking advantage of the user interface toolkits available in the Java platform.

23. Can I run standard syntax COBOL programs at the PDA/CDC level with isCOBOL APS?

Yes. isCOBOL APS, being Java-based, takes COBOL portability to a new level. You can run isCOBOL compiled programs on any system where the proper Java Runtime Environment release level is installed. Since the isCOBOL production environment executes in a Java platform, one can develop and deploy applications on desktops and servers, as well as embedded and real-time environments.