Integrate COBOL and Web-based business processes with a service-oriented architecture.

Enterprise COBOL for z/OS, Version 4 Release 2
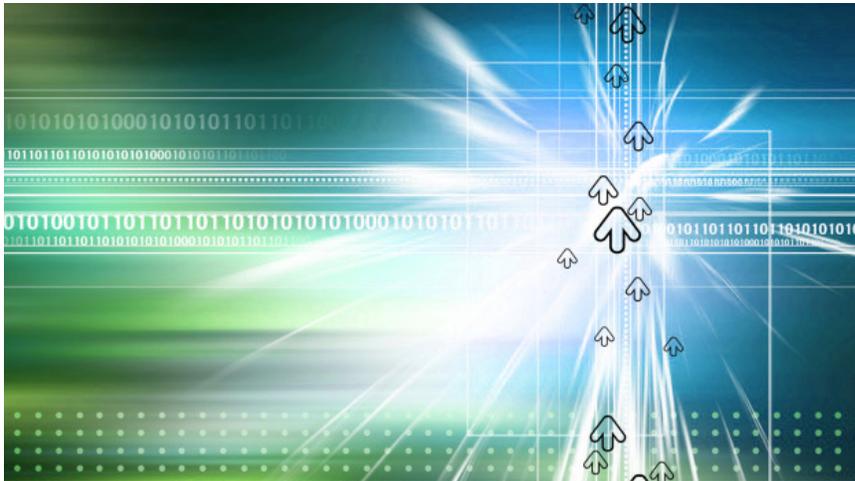
Highlights

- **Delivers enhanced XML parsing support**

- **Exploits system-determined block size for QSAM files, using the BLOCK0 compiler option**

- **Provides compiler listings that display CICS® options in effect**

- **Facilitates compiler message severity customization**

- **Supports the underscore (_) character in COBOL user-defined words**

- **Supports Java™ 5 and Java 6 SDKs for Java interoperability**

To make your business as agile and responsive as possible, you need to be able to connect your business components end to end with your suppliers, partners, employees, and customers, and you need to position your organization to quickly take advantage of opportunities by responding to challenges in real time.

Unfortunately, many IT systems weren't designed to address these objectives or to support Web services and service-oriented architecture (SOA) that are essential for transforming an enterprise into a flexible business with an open, integrated operating environment. You could rewrite your applications in a different programming language in order to address these objectives, but rewriting your applications would be expensive and risky, and it could potentially create downtime that you just can't afford. To remain competitive, you need a complete business strategy to help you modernize, integrate, and manage existing applications, data, and skill sets to ease your organization's transformation into a more flexible business.

Integrates COBOL applications with Web services

With Enterprise COBOL, you can take advantage of more than 40 years of IBM® experience in application development to integrate COBOL with Web services, XML, and Java. Such interoperability enables you to capitalize on existing IT investments while smoothly incorporating new, Web-based applications into your organization's infrastructure. One of the hottest new technologies for Enterprise COBOL is XML, described below.

Delivers enhanced XML parsing support

In Enterprise COBOL for z/OS®, Version 4.2, XML parsing using the z/OS XML System Services parser has been enhanced:

- You can now parse XML documents with validation against an XML schema, using the VALIDATING phrase of the XML PARSE statement.
- Performance is improved for nonvalidating parsing.
- Character reference processing is enhanced.

Enterprise COBOL for z/OS, Version 4.2 continues the support for the z/OS XML System Services parser, which provides:

- Offloading of COBOL XML parsing to IBM System z® Application Assist Processor (zAAP) specialty processors.
- Improved support for parsing XML documents that use XML namespaces.
- Direct support for parsing XML documents that are encoded in UTF-8 Unicode.
- Parsing of very large XML documents, one buffer of XML at a time.

Facilitates compiler message severity customization

Have you ever wanted to change the severity of a COBOL compiler diagnostic message, or even suppress one completely? In Enterprise COBOL for z/OS, Version 4.2, you can do both! For example, you can now get return code zero from a compilation that used to get warning messages.

The new MSGEXIT suboption of the EXIT compiler option lets you specify a module that is called for each compiler diagnostic message and each FIPS (FLAGSTD) message. Using the MSGEXIT module, you can change the severity of messages, suppress messages, and convert FIPS messages into diagnostic messages.

Exploits system-determined block size for QSAM files, using the BLOCK0 compiler option

In Enterprise COBOL for z/OS, Version 4.2, a new compiler option, BLOCK0, changes the compiler default for QSAM files from unblocked to blocked to gain the benefit of system-determined block size for output files. When a program is compiled using the BLOCK0 compiler option, an implicit BLOCK CONTAINS 0 clause is activated for all eligible QSAM files in the program, which can result in enhanced processing speed and minimized storage requirements for output files.

Supports the underscore (_) character in COBOL user-defined words

In Enterprise COBOL for z/OS, Version 4.2, COBOL user-defined words such as data names and program names can now include underscore characters (_). Underscores are also supported in the literal form of program names. Allowing underscores enables interoperability with XML, DB2/SQL, and other programming languages.

Provides compiler listings that display CICS options in effect

In Enterprise COBOL for z/OS, Version 4.2, if COBOL applications with embedded CICS statements are compiled using the integrated CICS translator, compiler listings show the CICS options that are in effect. This facility provides the same benefit to CICS users as was previously made available to DB2® users. You can now compile a DB2/CICS program in a single job step and get a single listing that includes COBOL, DB2, and CICS options in effect.

Supports Java 5 and Java 6 SDKs for Java interoperability

In Enterprise COBOL for z/OS, Version 4.2, Enterprise COBOL applications using object-oriented syntax for Java interoperability can now run with Java 5 or Java 6. Java SDK 1.4.2 continues to be supported. Enterprise COBOL supports object-oriented syntax to facilitate the interoperation of COBOL and Java. This support is based on the facilities of the Java Native Interface (JNI), the primary means that Java provides for interoperating with non-Java programs. Object-oriented COBOL is designed to enable you to:

- Define classes, with methods and data implemented in COBOL.
- Define classes that inherit from Java classes or from other COBOL classes.
- Create object instances of Java or COBOL classes.
- Invoke methods on Java or COBOL objects.
- Define and invoke overloaded methods.

Special register JNIENVPTR and the sample copybook JNI.cpy let you easily call services provided by the JNI, including services for handling EBCDIC and Unicode strings and for managing local and global object references.

Object arrays as method arguments

Enterprise COBOL for z/OS supports the use of Java object arrays. Object references of type jobjectArray can be specified as:

- Arguments in INVOKE…USING
- Values in INVOKE…RETURNING
- Parameters in the PROCEDURE DIVISION…USING phrase of method definitions
- Values in method definitions in PROCEDURE DIVISION…RETURNING

COBOL class definition with main method

In Enterprise COBOL for z/OS, an application that uses object-oriented COBOL syntax can start with a COBOL class definition. You can run the application by using the `java` command and pass string values as command-line arguments. You can also use this technique in other environments (such as Java regions under IMS™) that require applications to start with the main method of a Java class file.

Implementation of object-oriented COBOL programs from batch JCL

In certain cases, you can bind COBOL programs that use object-oriented syntax as modules in library (PDSE) data sets and run them using batch job control language (JCL).

Java interoperability and subsystems

Smooth interoperability between Java and COBOL in IMS

Enterprise COBOL supports interoperation between COBOL and Java under IMS when applications run in a Java dependent region. You can use Java to develop the messaging portion of an application and call COBOL to access IMS data resources. You can also develop object-oriented COBOL applications that contain a main method capable of invoking Java.

IBM WebSphere® Application Server interoperability

You can use the Java interoperability capabilities of COBOL to access enterprise beans that run on a Java Platform, Enterprise Edition (Java EE) technology–compliant Enterprise JavaBeans™ (EJB) server, such as IBM WebSphere Application Server. To do this, the client environment must support a Java technology–based object request broker (ORB). The client COBOL application can use COBOL INVOKE statements to access the following programming interfaces:

- Java Naming and Directory Interface (JNDI) to locate EJB services and components
- Java ORB interface to invoke methods on enterprise beans

Improved application development

Enterprise COBOL for z/OS provides a set of intrinsic functions including string handling, financial capabilities, statistical functions, and mathematical formulas. You can also use the COBOL CALL statement to take advantage of Language Environment® services for everything from storage management to condition handling. The condition-handling support enables you to write programs in which exception handling is done in a separate routine that is loaded only when needed. Using Language Environment condition handling, you do not have to write the exception-handling routines in assembler—you can write them in COBOL! Enterprise COBOL for z/OS offers support for recursive calls, structured programming, improved interoperability with other languages, and dynamic link library (DLL) support. The Enterprise COBOL for z/OS runtime library, Language Environment, also supports PL/I, C/C++, and Fortran programs.

Ease into migration

Enterprise COBOL for z/OS gives you a migration path from OS/VS COBOL, VS COBOL II, IBM COBOL for MVS & VM, and IBM COBOL for OS/390® & VM. With the exception of OS/VS COBOL programs and any programs that were previously compiled with the CMPR2 compiler option, your current programs can continue to compile and run without modification, while you selectively update existing applications to take advantage of new functions.

You can convert OS/VS COBOL programs and programs compiled with the CMPR2 compiler option into 1985 Standard programs, which can then be compiled using Enterprise COBOL for z/OS. Use the COBOL conversion tool (CCCA) included in Debug Tool Utilities and Advanced Functions for this purpose. Debug Tool Utilities and Advanced Functions also includes a load module analyzer that can help identify which of your programs were compiled with the OS/VS compiler.

Workstation-based development

Rational® Developer for System z provides an interactive, workstation-based environment to help you create, maintain, and reuse applications. Rational Developer for System z includes support for traditional development using COBOL, but also has the ability to generate Web services interfaces from COBOL constructs to ease creation of Web services from existing COBOL applications.

Rational Developer for System z provides a workstation interface to Debug Tool, and is also integrated with IBM File Manager and Fault Analyzer. File Manager integration enables you to access Keyed Sequence Data Set (KSDS) files from the Rational Developer for System z workbench, and gives you the ability to browse and update data sets. By integrating with Fault Analyzer, Rational Developer for System z enables you to browse Fault Analyzer ABEND reports on CICS, IMS, batch, Java, WebSphere, and other run times.

Rational Developer for System z, shown here, supports Enterprise COBOL and helps improve the productivity of COBOL developers. The windows displayed show the context-sensitive editor, as well as a compiler listing that indicates errors from a compilation. A simple click on a diagnostic message takes you to the line of source code in error.

COBOL across platforms

Enterprise COBOL for z/OS is part of a family of compatible compilers, application development tools, and maintenance tools. Along with Enterprise COBOL for z/OS, IBM offers COBOL compilers for multiple platforms as well as IBM File Manager, IBM Fault Analyzer, and Debug Tool. As mentioned previously, the recommended workstation-based development environment is Rational Developer for System z.
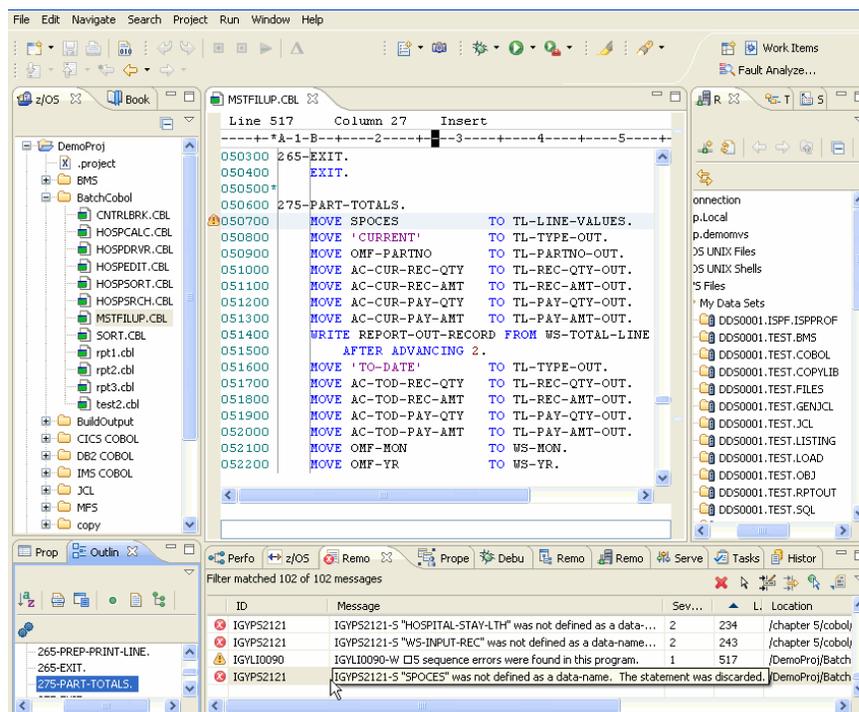
Figure 1: Rational Developer for System z workbench.

For more information visit
www.ibm.com/software/awdtools/cobol/zos

SC23-8612-00

IBM