What's new in COBOL for Linux on x86 1.2

COBOL for Linux® on x86 1.2 and subsequent fix packs that are noted with version tags add support for the following new features. If you are migrating COBOL source programs from COBOL for Linux on x86 1.1 or another COBOL compiler to COBOL for Linux on x86 1.2, see the *Migration Guide* for differences between compilers.

COBOL/ Java Interoperability

1.2.0.8+

COBOL for Linux on x86 1.2 supports seamless data and control exchange between COBOL and Java programs. During compilation, the compiler generates intermediate files that use the Java Native Interface (JNI) to call native routines and convert data between environments.

Loading the dynamic shared object created by these intermediate files enables the Java applications to call user-defined COBOL programs. Similarly, COBOL programs can call Java methods by using the standard `CALL` statement.

For more information on COBOL/Java Interoperability, see Programming for a Java environment.

New Compiler Directive

1.2.0.8+

Use the `JAVA-CALLABLE` directive to enable COBOL programs to be called from Java. This directive instructs the compiler to generate intermediate files that handle data conversion and pass control from Java to the native COBOL program. The compiler links these files into a DSO that Java can load.

For more information, see JAVA-CALLABLE Directive.

Integration with PostgreSQL

1.2.0.6+

COBOL for Linux on x86 1.2 integrates with PostgreSQL, an open-source database, through SQL statements embedded in COBOL programs. This integration provides a PostgreSQL coprocessor instead of a preprocessor, which means that the compiler translates the EXEC SQL statements into COBOL statements during the compilation process itself, rather than during a separate step before compilation.

For more information about integration with PostgreSQL, see https://ibmdocs-test.dcs.ibm.com/docs/en/cobol-linux-x86/1.2.0?topic=environments-programming-postgresql-environment.

New options

- 1.2.0.8+

  The compiler uses the `JAVAPARA` option to control how arguments are exchanged between COBOL and Java applications. This option accepts two values, `obj` and `barr`.

  - Use `obj` to indicate that arguments are passed as Java objects.
  - Use `barr` to indicate that arguments are passed as byte arrays.

  For more information, see JAVAPARA.

- 1.2.0.8+

  Use the `JAVAPKG` option to control the package name for generated Java Native and Java Accessor stubs. By default, the compiler assigns the package name `com.ibm.cobol.javaiop` to these stubs.

  For more information, see JAVAPKG.

- 1.2.0.6+

The SQL(PGSQL) option is added for building programs that work with PostgreSQL. This option tells the compiler that PostgreSQL is the database management system for processing all SQL statements in the COBOL program.

For more information about programming for a PostgreSQL environment, see https://ibmdocs-test.dcs.ibm.com/docs/en/cobol-linux-x86/1.2.0?topic=environments-programming-postgresql-environment.

- 1.2.0.2+

  The ARITH(FULL) option is added with the fix pack for APAR PH58656 installed. ARITH(FULL) is the same as ARITH(EXTEND) except for the number of decimal digits preserved in intermediate results. For more information about the ARITH option, see ARITH.

Compiler invocation command cob2_pgsql

1.2.0.6+

The **cob2_pgsql** compiler invocation command is added for working with PostgreSQL. The command automatically adds the required compiler and linker options. You can customize the behavior of the **cob2_pgsql** command by modifying the compiler configuration file, cob2.cfg. For example, you can specify a different location for the ECPG library, or, adjust the path to the version of PGSQL in use.

For more information about customizing the configuration file, see Modifying the default compiler configuration.

64-bit application development support

1.2.0.0+

COBOL for Linux on x86 1.2 provides support for creating 64-bit COBOL applications. This provides access to a large address space for code and application data, more efficient use of memory layout, and improved performance.

You can control whether 64-bit or 32-bit COBOL programs are created through use of the ADDR compiler option. By default, the compiler will create 64-bit applications. For more information about the ADDR compiler option, see ADDR.

In 64-bit mode, storage allocation for data items that contain addresses or indexes (POINTER, FUNCTION-POINTER, PROCEDURE- POINTER, and INDEX) is increased to 8 bytes, affecting data items that have any of these usages.

Additional benefits include the ability to link COBOL applications with 64-bit C/C++ and other compiled language applications, and to inter-operate with 64-bit Java™ applications through JNI. COBOL programs that make use of IBM® Db2® might see improved performance as the program no longer needs to spend cycles transitioning data between a 64-bit Db2 server and a 32-bit Db2 client.

The debugger included with COBOL for Linux on x86 1.2 provides the capability to debug 64-bit COBOL applications.

Incremental build support

1.2.0.0+

Incremental build support is provided through the -M compiler option. When the -M compiler option is specified, the compiler will generate dependency information that can be used in a makefile. GNU makes use of the dependency information and will automatically recompile a COBOL program if its source or any of the copybooks it uses are modified. This may help improve developer productivity by allowing iterative, incremental builds, while also ensuring that programs are always built with the latest version of any copybooks that are used.

For more information about the -M compiler option, see -M.

MongoDB as a VSAM data store

1.2.0.0+

Support for MongoDB as a VSAM data store is provided by specifying a file system type of MONGO in COBOL applications.

For more information about MongoDB, see Using MongoDB files.

SLES 15 support

1.2.0.0+

SLES 15 support is available for building and deploying 64-bit COBOL applications. SLES 15 only supports building and deploying of 64-bit COBOL applications. For building and deploying 32-bit COBOL applications, use either RHEL 8 or 9 or Ubuntu 20.04 or 22.04, as these platforms provide the ability to build and deploy both 32-bit and 64-bit COBOL applications.

For more information about the system requirements of COBOL for Linux on x86 1.2, see System prerequisites.