IBM z/OS V2R1 XL C/C++

Enable high-performing z/OS XL C/C++ programs for workload optimized business software solutions

Highlights

- Exploits zEnterprise® System servers through new hardware built-in functions and options for improved application performance
- Enables system programming capabilities through Metal C with support for advanced optimization for AMODE-switching Metal C applications and programming features to reduce system programming effort
- Supports additional C11 and C++11 features, and GNU C/C++ compatibility features for easier programming and better portability
- Provides performance optimization with support for a new option to specify single-threaded programs, OpenMP parallelization directives, and changes in the default ARCH/TUNE settings
- Enhances debugging for optimized code and inline procedures
- Integrates with IBM® Rational® Developer for System z® and IBM Rational Team Concert™ providing a modern development environment and a collaborative team environment

The IBM z/OS® XL C/C++ compiler helps you create and maintain critical business applications written in C or C++, maximize application performance, and improve developer productivity. z/OS XL C/C++ can transform C or C++ source code to fully exploit your existing IBM System z hardware and optimize workloads through smarter computing capabilities with the new IBM zEnterprise System. Built-in functions, performance-tuned libraries, and language constructs are some of the features that simplify programming and boost application runtime performance.

IBM works constantly to improve compiler components, including front-ends, high-level optimizers, and low-level optimizers. By upgrading your compiler, you can keep up with new language standards and extensions, advancements in hardware technology, usability features, and advances in optimization with minimal or no source code changes. IBM compilers offer a cost-effective way to get more out of existing technology and stay ahead of competitors on the technology curve.

z/OS XL C/C++ is a leading-edge compiler that maximizes middleware by providing access to IBM DB2®, CICS®, and IMS™ systems.

This new release of z/OS V2R1 XL C/C++ reinforces the continuing IBM commitment to the C and C++ programming languages on the z/OS platform.

Exploits zEnterprise System servers through new built-in functions and options

In z/OS V2R1 XL C/C++, the ARCH(10) option is introduced, to enable functions for programs running on zEnterprise EC12 (zEC12) and zEnterprise BC12 (zBC12) systems, including support for the miscellaneous-instruction-extension facility, and the transactional-execution facility, as well as a new TUNE(10) option to generate code that is optimized for zEC12 and zBC12 machines.

New built-in functions have been added to mark the beginning and end of transactions, and to diagnose the reasons for failure. These built-ins include functions for starting, ending and aborting a transaction, as well as for storing non-transactional state, extracting the nesting depth of a transaction, and for capturing transaction failure diagnostics. The built-ins provide an effective means for improving the performance of multithreaded C and C++ applications on z/OS.

New hardware built-in functions are also available for accessing the Decimal-Floating-Point Zoned-Conversion facility in zEC12 and zBC12 which enables zoned-decimal types to take advantage of the DFP unit to increase application performance and CPU utilization.

Enables system programming capabilities through Metal C

System programming allows for development of freestanding programs that do not depend on any supplied runtime environment, such as Language Environment®. These programs obtain the system services needed by calling assembler services directly.

You can use the Metal C feature of z/OS XL C in place of assembler for development of system programs. Metal C includes the following capabilities:

- The __asm keyword enables you to specify assembly instructions in C code.
- The METAL compiler option enables you to use the MVS™ system linkage conventions.
- The PROLOG and EPILOG compiler options and the #pragma prolog() and #pragma epilog() preprocessor directives enable you to specify custom HLASM prolog and epilog code, to implement custom function linkage conventions if required.

System programs written in C exploit the same advancements in hardware and advances in performance optimization realized by application programs coded in C.

Metal C can also be used to write programs to run in the CICS Transaction Server for z/OS, particularly in situations where you would otherwise code directly using assembler.

Improves the optimization of Metal C programs through advanced optimization for AMODE-switching Metal C applications

New support for AMODE switching during IPA link enables Metal C applications that use AMODE switching to take advantage of interprocedural analysis optimizations.

New #pragma map directive to associate the "main" function to an alternate external name

When a Metal C program is built with the RENT option, it needs a "main" function to anchor the writable static area (WSA) creation process. However, a Metal C program may not have a function called "main" as the entry point thus not having the opportunity to be built with the RENT option.

A Metal C program can now have an alternate entry point name for function "main" while maintaining all the characteristics of function "main".

New SYSSTATE option to enable Metal C users to fine tune the SYSSTATE assembler macro

The SYSSTATE assembler macro automatically generated by the compiler does not have the OSREL parameter nor the ASCENV parameter specified.

The new SYSSTATE option allows the user to add the OSREL parameter with the desired value, and to have the ASCENV parameter generated according to the ASC mode of the function.

Supports additional C11 and C++11 features, and GNU C/C++ compatibility features for easier programming and better portability

z/OS V2R1 XL C/C++ supports additional C11 and C++11 functions to allow a standardized way to specify optimization choices, make it easier to write C and C++ programs, improve program portability, and help you with debugging. The compiler also adds support for additional GNU C/C++ language extensions and compatibility features for ease of migrating applications built with GNU C/C++ to System z.

C11 features

z/OS V2R1 XL C/C++ supports the following C11 features:

Anonymous structures
The z/OS V2R1 XL C/C++ compiler enables anonymous structures under the EXTC1X extended language level. An anonymous structure is a structure that does not have a tag or a name and that is a member of another structure or union.

Complex type initialization
When the C11 complex initialization feature is enabled, you can initialize C99 complex types with a value of the form x

+ yi, where x and y can be any floating point value, including Inf or NaN.

Generic selection
: Generic selection provides a mechanism to choose an expression according to a given type name at compile time, a common usage of which is to define type generic macros.

_Noreturn function specifier
: When you declare a function with the _Noreturn function specifier, the compiler can better optimize your code without regard to what happens if it returns. Any function, with the exception of main, can be declared or defined with the _Noreturn function specifier.

Static assertions
: You can use static assertions to detect and diagnose common usage errors at compile time.

C++11 features

z/OS V2R1 XL C/C++ supports the following C++11 features:

Default and deleted functions
: This feature introduces two new forms of function declarations to define explicitly defaulted functions and deleted functions. For the explicitly defaulted functions, the compiler generates the default implementations, which are more efficient than manually programmed implementations. The compiler disables the deleted functions to avoid calling unwanted functions.

Explicit conversion operators
: With this feature, you can apply the explicit function specifier to the definition of a user-defined conversion function to inhibit unintended implicit conversions through the user-defined conversion function.

Generalized constant expressions
: With this feature, constant expressions can include calls to template and non-template constexpr functions, constexpr objects of class literal types, and references bound to const objects that are initialized with constant expressions.

Strongly scoped enums
: With the scoped enumeration feature, you can get the following benefits:

- The ability to declare a scoped enumeration type, whose enumerators are declared in the scope of the enumeration.
- The ability to declare an enumeration without providing the enumerators.
- The ability to specify explicitly the underlying type of an enumeration.
- Improved type safety with no conversions from the value of an enumerator (or an object of an enumeration type) to an integer.

rvalue references
: With the rvalue references feature, you can reuse the resources of expiring objects and improve the performance of your libraries, especially if you use generic code with class types. Additionally, the value category can be considered when writing a forwarding function.

Right angle brackets
: In the C++ language, two consecutive closing angle brackets (>) in the template parameter context must be separated with a white space, because they are otherwise parsed as the bitwise right-shift operator (>>). The right angle bracket feature removes the white space requirement for consecutive right angle brackets, thus making programming more convenient.

GNU C/C++ compatibility

z/OS XL C/C++ V2R1 supports the following GNU C/C++ compatibility features:

- __builtin_expect
- propagation of attributes to function template instantiations
- zero initialization for objects with an initializer of () and an implicitly defined default constructor
- improved diagnostics for invalid template template argument

Provides optimization with support for a new option to specify single-threaded programs, OpenMP parallelization directives, and change in the default ARCH/TUNE settings

The z/OS XL C/C++ compiler assumes that the user application is multithreaded, which inhibits it from making non-threadsafe transformations which could speed up code. In V2R1, users can use the new NOTHREADED option for their single-threaded applications for improved compile time and run time performance benefits.

OpenMP is a widely used industry standard to construct shared memory parallelism. The new SMP option allows OpenMP parallelization directives to be recognized.

In V2R1, the default ARCH and TUNE settings have been changed to target IBM System z9® servers, and above, for improved application performance. The default settings are changed from ARCH(5) and TUNE(5) to ARCH(7) and TUNE(7) in z/OS V2R1.

Enhances debugging for optimized code and inline procedures

z/OS V2R1 XL C/C++ introduces new debug level options that make it easier to generate optimized code that can still be easily debugged.

In addition, the compiler offers debug information for parameters and local variables of each inline instance of a procedure.

Integrates with IBM Rational Developer for System z and IBM Rational Team Concert providing a modern development environment and a collaborative team environment

IBM Rational Developer for System z, an Eclipse-based offering, boosts programming productivity with an integrated development environment that makes it easy to edit, compile and debug z/OS XL C and XL C++ applications right from your workstation.

IBM Rational Team Concert allows you to boost programming productivity with a collaborative team environment that makes it easy to manage your distributed software projects and teams.

License options

To help you optimize software licensing costs, IBM can assist in identifying the licenses that best suits your organization. For additional information on the types of licenses available for z/OS, see:

www.ibm.com/systems/z/resources/swprice/index.html

Ordering information

IBM z/OS XL C/C++ is an optional priced feature of z/OS. z/OS XL C/C++ is available through the ShopzSeries web site:

www.ibm.com/software/shopzseries

where it is listed as “XL C/C++”.

z/OS XL C/C++ is supported on z/OS at the same level. For more information about the support lifecycle of z/OS, see:

www.ibm.com/software/support/lifecycle/index_z.html

Summary of features and benefits

The following table summarizes the features and benefits for z/OS XL C/C++.

Table 1. Summary of features and benefits

Feature	Benefit
Designed for IBM platforms	• Exploits z/Architecture® systems
Improved industry language standards compliance	• Facilitates porting from other platforms to z/OS • Provides compiler diagnostics to help you achieve the level of conformance to a particular programming language standard • Supports commonly used IBM and non-IBM language extensions
Improved industry-leading optimizations	• Uses technology from the industry-leading XL C/C++ family of compilers; z/OS XL C/C++ is designed to offer superior application performance on z/OS.
Enhanced middleware support	• Exploits the latest middleware (DB2, CICS, IMS) to facilitate application integration and modernization.
Improved debug information and debug APIs	• Enables tool providers (including Debug Tool, Application Performance Analyzer, z/OS dbx, as well as third-party tools) to build additional debugging capability and improve performance in their tools.
Improved low-level programming support	• Provides system programming capabilities through Metal C. With Metal C you can insert HLASM instructions into C source, specify custom function prologs and epilogs, and generate HLASM source, making it easier to integrate new code with existing HLASM programs.
Exploits hardware support for IEEE 754 decimal floating-point data	• Improves the accuracy and performance of decimal floating-point calculations for commercial applications
Additional built-in functions	• Provides access to the newest and most efficient hardware operations at the source level • Simplifies the development effort for creating and maintaining high-performance applications
Integrated development environment	• Rational Developer for System z consists of a common development workbench and an integrated set of tools that support model-based development, runtime testing, and rapid deployment of applications.
Collaborative team environment	• Rational Team Concert (separate product) unifies development teams by making it easy to manage your distributed software projects and teams.
IBM service and support	• Provides responsive platform and cross-platform support that meets or exceeds customer expectations. • Teams with subject matter experts in compiler development for dedicated support excellence.

System requirements

The following table presents the hardware and software requirements for z/OS V2R1 XL C/C++.

Table 2. System requirements

Operating system	**Software**	**Hardware**
z/OS	V2R1	• IBM zEnterprise EC12 (zEC12) • IBM zEnterprise BC12 (zBC12) • IBM zEnterprise 196 (z196) • IBM zEnterprise 114 (z114) • IBM System z10™ (z10 EC, z10 BC)[1] • IBM System z9 (z9 BC, z9 EC)[1] **Notes:** 1. These products are withdrawn from marketing.

Upgrade now

Upgrade to the latest z/OS operating system and get the latest XL C/C++ compiler to leverage your zEnterprise investment and stay ahead of competitors on the technology curve.

For more information

To learn more about z/OS V2R1 XL C/C++, contact your IBM representative or IBM Business Partner, or visit the z/OS XL C/C++ web site at http://www.ibm.com/software/products/us/en/czos.

Like IBM Compilers on Facebook or follow @IBM_compilers on Twitter. The IBM C/C++ Cafe at http://www.ibm.com/rational/community/cpp has additional resources on IBM C and C++ compilers.