



Application Compatibility Guide

Micro Focus
The Lawn
22-30 Old Bath Road
Newbury, Berkshire RG14 1QN
UK
<http://www.microfocus.com>

© Copyright 2019 Micro Focus or one of its affiliates.

MICRO FOCUS, the Micro Focus logo and Visual COBOL and Enterprise Developer are trademarks or registered trademarks of Micro Focus or one of its affiliates.

All other marks are the property of their respective owners.

2019-06-19

Contents

Application Compatibility Guide	4
Compatibility Overview	5
PL/I Application Compatibility	6
COBOL Language Compatibility	7
Third-party Compatibility	8
SOA/IMTK Interface Definition Table (IDT) Compatibility	9
Visual COBOL and Enterprise Developer Windows C Runtime Compatibility	10
Visual COBOL and Enterprise Developer UNIX and Linux C Runtime Compatibility	13
Glossary of Definitions	17

Application Compatibility Guide

If you are planning to upgrade or update to a newer Micro Focus product, then you will be giving some thought to how your existing applications will be affected.

Micro Focus supports the forward compatibility of callable artifacts built with one version of a product, and then deployed and executed on a later version of the product running on the same platform, and in the same Product Family, unless explicitly specified in the release documentation. The same may also be true for main executable programs (.exe files), although these files have an extra dependency on the version of the C runtime associated with the products used to build and to run the application.

This guide aims to demonstrate exactly which products are compatible with each other, and whether or not you can just deploy your existing applications on a newer product without the need to recompile or relink them. This guide covers recommendations for both COBOL and PL/I applications.

Use the tables and diagram in the Runtime Compatibility sections of this document to determine exactly which products are compatible with each other.



Note: Some of the older products in these tables are now out of support. Check the Product Availability section on the Micro Focus SupportLine Web site - <https://supportline.microfocus.com/prodavail.aspx> - for more information.

Compatibility Overview

On a Microsoft Windows platform, when building to a main COBOL executable (.exe), compatibility depends on the Microsoft C runtime associated with the products used to build and run the executable. The executable is compatible with any future COBOL products that share a common C runtime with the product on which it was built. See the *Visual COBOL and Enterprise Developer Windows C Runtime Compatibility* sections for details on C runtimes used with COBOL products on Windows.

However, if the application has a main COBOL executable, and if the product being used for deployment is based on a C runtime that is later than that in the product used for building the application, then compatibility cannot be guaranteed. If the behavior of the application changes when you run with the later version then Micro Focus strongly recommends that the main executable is relinked with the later product. This will ensure that the COBOL runtime fully handles any runtime error conditions that may occur.

Relinking (to the new run time system) enables you to create a new executable capable of running on the later product, without the need to recompile. To relink, you must use the object code that was generated when the executable was originally compiled.

On UNIX and Linux, COBOL products are built with specific C and C++ compilers, and tested against specific versions of the C and C++ runtime. In general, COBOL application artifacts can be executed with later versions of the C and C++ runtime, unless the C/C++ vendor has specified otherwise; see the *Visual COBOL and Enterprise Developer UNIX and Linux C Runtime Compatibility* section for more details. The compatibility of `glibc` versions is similar in nature to a third-party component, such as a Relational Database Management System or Java Application Server, so version checking is advised.

Micro Focus does not test all application deployment combinations, but will support valid configurations and respond to reported incidents if incompatibilities are found, and then make reasonable commercial efforts to provide a solution. However, in some of these circumstances, recompiling with a later version might be the required action.

There are other situations in which recompilation or relinking are the only valid courses of action; for example, debug and diagnostic output from a mixed deployment environment can only be processed with the corresponding latest development product that matches the deployment environment. Changes in platform could also require the application to be modified, recompiled or relinked if external components introduce incompatibilities.

When compiling and running managed COBOL applications, applications compiled for .NET managed code contain references to specific versions of the .NET framework and the COBOL runtime assemblies, but in general, backward compatibility is provided in updated versions. COBOL applications compiled for JVM managed code do not specifically target later Java versions, and in general, will be compatible when using a supported Java version.

PL/I Application Compatibility

PL/I applications currently have more restrictive compatibility support. It is recommended that PL/I applications are rebuilt on every release, to ensure application execution behavior is compatible with the PL/I runtime.

COBOL Language Compatibility

The syntactical and semantic definition of the COBOL language is defined by ANSI, and has been extended by different vendors such as IBM and Micro Focus. Emulation of a particular standard or vendor version can be controlled by the use of compiler directives and runtime configuration.

The COBOL language definition refers to non-standard usage as having either undefined behavior or undefined results - any application that relies upon this syntax can be subject to changes in behavior between product versions when recompiled.

Occasionally, incompatibilities or defects are discovered and corrected. Applications that rely on previously incorrect behavior can also be subject to change in later products when recompiled.


Upgrading to the latest product on a regular basis will help identify any incremental incompatibilities or reliance on undefined behavior, making your applications more portable and less prone to changes between subsequent versions. If there is an extended interval between upgrade and recompilation, then it increases the risk of application remediation work being required.


If you want to take advantage of new product capabilities, such as performance improvements or language extensions, then recompilation on the latest product will be required.

Third-party Compatibility

Historically, Micro Focus has indicated that an operating system or third-party component is supported once testing has been completed on that version against a reference COBOL product.

The following extended classification of third-party components provides more guidance for customers who are planning to upgrade to a component before Micro Focus officially classifies it as supported.

Stage	Commitment
Planned	<ul style="list-style-type: none"> Will apply to updated versions of operating systems or third-party components where Micro Focus already supports an earlier version. Customers can report incompatibility issues, and commercial best-efforts will be made to provide a resolution, but no commitment before full Tested state, which may be the next release.
Tested	<ul style="list-style-type: none"> Appropriate levels of testing have been performed and Micro Focus can state full support for a given operating system or third-party component.
Previously Tested	<ul style="list-style-type: none"> Typically applies to products that have reached EOS, that Micro Focus previously tested, but no longer tests. <p> Note: If the vendor no longer supports the operating system or third-party component, Micro Focus is under no obligation to resolve issues.</p>
Not Currently Planned	<ul style="list-style-type: none"> Covers products where Micro Focus either predicts or knows about a compatibility problem with no known resolution.
<Not Specified>	<ul style="list-style-type: none"> The absence of any status typically indicates that there is no support, and support is not currently planned.

 **Note:** **Planned** and **Tested** status means that Micro Focus will accept incident reports.

Micro Focus may also accept incidents for unsupported operating systems if those operating systems are fully-compatible with a supported operating system. In such circumstances, the incident would be tested on the supported operating system, and if the behavior is consistent, it can be investigated. If the problem is believed to be non-Micro Focus related (e.g. an environment issue), you must refer to the vendor for support.

SOA/IMTK Interface Definition Table (IDT) Compatibility

The IDT is both forwards- and backwards-compatible: that is, the IDT-related behavior is dependent on the product used to build the IDT and the product version used to execute the program that includes the IDT. For example, the latest Visual COBOL product can handle old Net Express IDTs in the same way as Net Express did. Similarly, an IDT created with Visual COBOL should theoretically be usable by Net Express and give the same behavior as if it had been created with Net Express. However, to benefit from the latest behavior that is available, the most recent version of Visual COBOL is always recommended for both creation and execution.

If you are using a Micro Focus Web or EJB client to communicate with a service running under Enterprise Server, then the most important thing is to always make sure the client and server are fully in sync (and so contain the same IDT). Problems can occur when customers have updated the client, and then either forgotten to re-deploy the service or thought they just didn't need to. Failure to do either of these will cause problems.

32- and 64-bit Compatibility

The IDT contains no information related to bitism and can generally be used for both 32- and 64-bit deployment without change.

The only exception to this is when the interface COBOL data contains an item that varies in size between 32- and 64-bit (such as a pointer or procedure-pointer item), as this may change the alignment of data fields - this is information stored in the IDT, which comes from the COBOL dictionary file. Up to Visual COBOL/Enterprise Developer 3.0, these pointer types could only be an issue if they existed in the original interface of a COBOL service being deployed in Enterprise Server, which admittedly is very rare. However, new support in Visual COBOL/Enterprise Developer 4.0 onwards will itself now use pointers in the interfaces, and so this may now become more of an issue.

Visual COBOL and Enterprise Developer Windows C Runtime Compatibility

The following tables show the version of the C Runtime that is used with each Windows product:

Product	Version	C Runtime
Visual COBOL for Visual Studio 2010	R2	VS2010
	R3	VS2010
	R4	VS2010
	R4U1	VS2010
	R4U2	VS2010
Visual COBOL for Visual Studio 2010 COBOL Server for Windows Enterprise Developer for Visual Studio 2010 Enterprise Server for Visual Studio 2010	2.0	VS2010
	2.1	VS2010
	2.1.1	VS2010
	2.2	VS2010
	2.2.1	VS2010
	2.2.2	VS2010
Visual COBOL for Visual Studio 2012 COBOL Server for Windows Enterprise Developer for Visual Studio 2012 Enterprise Server	2.1.1	VS2012
	2.2	VS2012
	2.2.1	VS2012
	2.2.2	VS2012
	2.3	VS2012
	2.3.1	VS2012
	2.3.2	VS2012
	3.0	VS2012
Visual COBOL for Visual Studio 2013 Enterprise Developer for Visual Studio 2013 COBOL Server for Windows Enterprise Server	2.2.1	VS2013
	2.2.2	VS2013
	2.3	VS2012
	2.3.1	VS2012
	2.3.2	VS2012
	3.0	VS2012
Visual COBOL for Visual Studio 2015 Enterprise Developer for Visual Studio 2015 COBOL Server for Windows Enterprise Server	4.0	VS2017
	2.3	VS2012
	2.3.1	VS2012
	2.3.2	VS2012
	3.0	VS2012
Visual COBOL for Visual Studio 2017 Enterprise Developer for Visual Studio 2017 COBOL Server for Windows Enterprise Server	4.0	VS2017
	3.0	VS2012
	5.0	VS2017
	4.0	VS2017
Visual COBOL for Visual Studio 2019 Enterprise Developer for Visual Studio 2019 COBOL Server for Windows Enterprise Server	5.0	VS2017
	5.0	VS2017
	5.0	VS2017

Product	Version	C Runtime
Visual COBOL for Eclipse 2010	R4	VS2010
	R4U1	VS2010
	R4U2	VS2010
Visual COBOL for Eclipse Enterprise Developer for Eclipse	2.0	VS2010
	2.1	VS2010
	2.1.1	VS2010
	2.2	VS2010
	2.2.1	VS2010
	2.2.2	VS2010
	2.3	VS2012
	2.3.1	VS2012
	2.3.2	VS2012
	3.0	VS2012
	4.0	VS2017
5.0	VS2017	

Recommendations

The optimum deployment configuration is to match the build and execution environment versions, as this combination has the most extensive compatibility testing.

To take advantage of the latest deployment product and its associated capabilities and features, such as platform support, performance, security and product features in a timely manner, you could employ a staged transition of deploying objects built with earlier development products.

The compatibility options are described within the following sections, and vary depending on: the products used, the format of the COBOL application artifacts, and the deployment platform and associated external components.

The following table shows the compatibility of each object type, if a staged transition was to be used:

Object Type	OS	Compatibility	Notes
.int	All	Platform	Slowest format, not shared
.gnt	All	Platform	Not shared
.dll	Windows	Product Family	Code shared between processes at OS level
.exe	Windows	Product C Library	Requires relinking when C library changes
.so	Unix/Linux	Product Family	Code shared between processes at OS level
<executable>	Unix/Linux	Product Family	Requires relinking when C vendor indicates


If you do not intend to keep the development and deployment product versions aligned, it is recommended that you keep the intermediate object code (.obj, .o) from the COBOL compiler and the linker command line options from the final build prior to deployment. If you subsequently upgrade the deployment product to one that requires the standalone executable to be relinked, then you can rebuild the executable, without having to recompile the application from source with the new COBOL compiler.

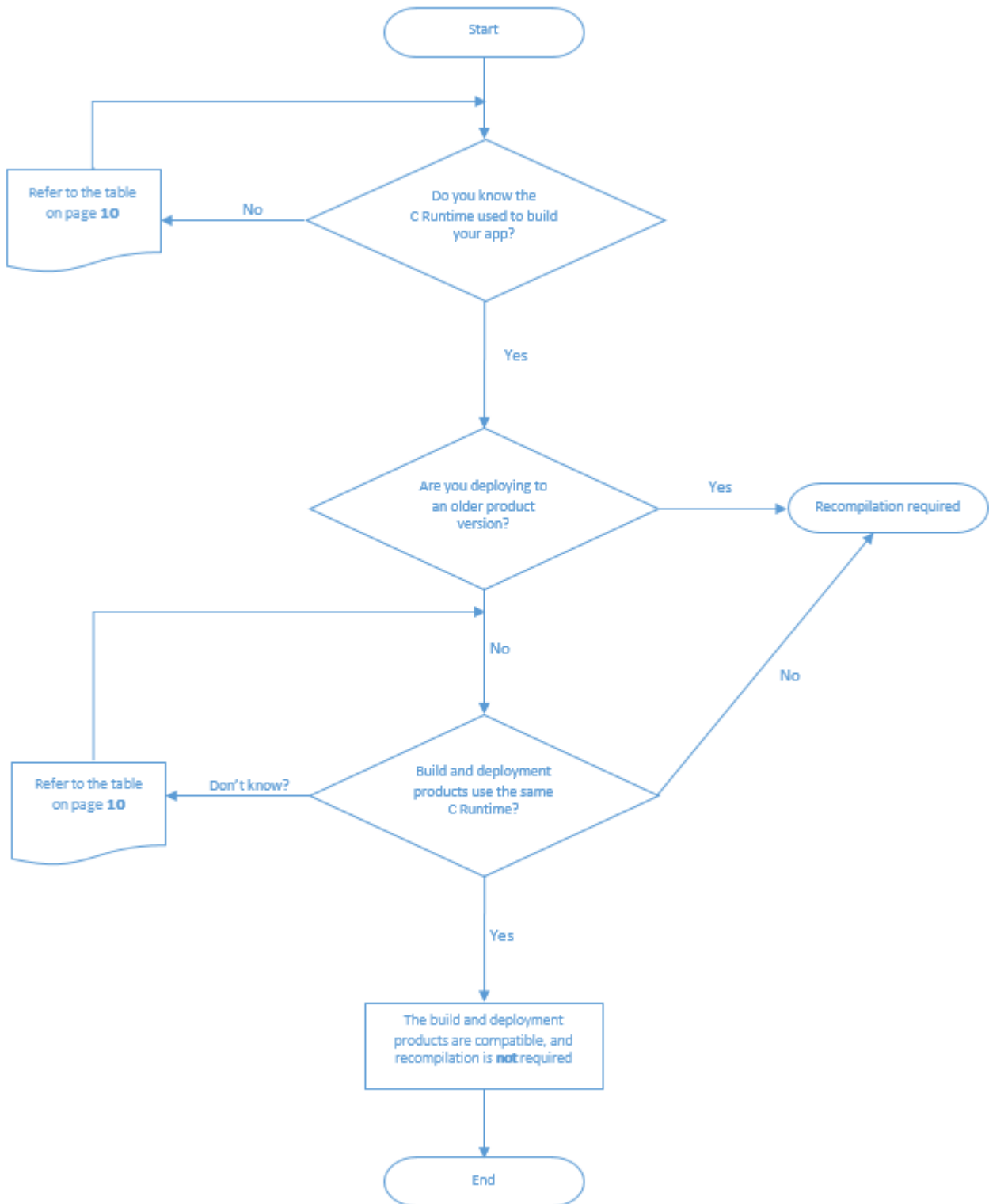
Compatibility between build and run products

The following sections show the compatibility between build and run products, on Windows platforms.

As previously mentioned, unless explicitly stated in your current release documentation, the callable artifacts of your application will be forward-compatible when running on the same platform, and using a later product, in the same Product Family, to the one in which they were originally built.

Use the following flow diagram to determine if you need to recompile when building your programs with one product, and then running them with another:

 **Note:** If you deploy to an enterprise server, a main executable is not deployed; therefore, you only need to ensure that your callable artifacts are compatible, using the criteria above.



Visual COBOL and Enterprise Developer UNIX and Linux C Runtime Compatibility


The following table shows the compatibility of Visual COBOL Development Hub and Enterprise Developer UNIX Components products on UNIX/Linux platforms, and their associated C runtime versions.

Product	Operating system	Version	C Runtime Version	Notes
Visual COBOL Development Hub 2.x Enterprise Developer UNIX Components 2.x	AIX	6.1 TL7 SP4	IBM xIC.rte 11.1.0.2	C Runtime (xIC.rte package) is independent of the OS install, so Micro Focus has chosen to install these versions on our reference test machines.
		6.1 TL9 SP8	IBM xIC.rte 13.1.3.1	
		7.1 TL4 SP3	IBM xIC.rte 13.1.3.1	
		7.2 TL1 SP1	IBM xIC.rte 13.1.3.1	
	HP	11i v3 (1603)	aC++ runtime A.06.28	C runtime is separate to the OS.
		11i v3 (1109)	aC++ runtime A.06.25.01	
	Oracle Linux RH Kernel	7.3	glibc 2.17	
	Oracle Linux Unbreakable Kernel	6.8	glibc 2.12	
		7.3	glibc 2.17	
	RHEL	6	glibc 2.12	
		6.2	glibc 2.12	
		6.8	glibc 2.12	
		7.1	glibc 2.17	
		7.3	glibc 2.17	
	Solaris	10	Solaris Studio 12.2	
		11	Solaris Studio 12.2	
		11.3	Solaris Studio 12.2	
	SUSE	11	glibc 2.9	
		11 SP1	glibc 2.11	
		11 SP4	glibc 2.11	
12		glibc 2.19		
12 SP2		glibc 2.22		
Visual COBOL Development Hub 3.0	AIX	7.1 TL4 SP3	IBM xIC.rte 13.1.3.1	C Runtime (xIC.rte package) is independent of the

Product	Operating system	Version	C Runtime Version	Notes
Enterprise Developer UNIX Components 3.0				OS install, so Micro Focus has chosen to install this version on our reference test machines.
	HP	11i v3 (1603)	aC++ runtime A. 06.25	C runtime is separate to the OS.
		11i v3 (1109)	aC++ runtime A. 06.25.01	
	Oracle Linux RH Kernel	7.3	glibc 2.17	
	Oracle Linux Unbreakable Kernel	6.8	glibc 2.12	
		7.3	glibc 2.17	
	RHEL	6	glibc 2.12	
		6.2	glibc 2.12	
		6.8	glibc 2.12	
		7.1	glibc 2.17	
		7.3	glibc 2.17	
	Solaris	10	Solaris Studio 12.2	
		11.3	Solaris Studio 12.6	
	SUSE	11	glibc 2.9	
		11 SP1	glibc 2.11	
		11 SP4	glibc 2.11	
		12	glibc 2.19	
12 SP2		glibc 2.22		
Visual COBOL Development Hub 4.0 Enterprise Developer UNIX Components 4.0	AIX	7.1 TL4 SP3	IBM xIC.rte 13.1.3.1	C Runtime (xIC.rte package) is independent of the OS install, so Micro Focus has chosen to install these versions on our reference test machines.
		7.2 TL2 SP2	IBM xIC.rte 13.1.3.1	
	HP	11iv3 (1703)	aC++ runtime A. 06.28	C runtime is separate to the OS.
		11i v3 (1109)	aC++ runtime A. 06.25.01	
	Oracle Linux RH Kernel	7.4	glibc 2.17	
	Oracle Linux Unbreakable Kernel	6.9	glibc 2.12	

Product	Operating system	Version	C Runtime Version	Notes
		7.4	glibc 2.17	
	RHEL	6	glibc 2.12	
		6.2	glibc 2.12	
		6.9	glibc 2.12	
		7.1	glibc 2.17	
		7.4	glibc 2.17	
	Solaris	10	Solaris Studio 12.2	
		11.3	Solaris Studio 12.6	
	SUSE	12 SP2	glibc 2.22	
		12 SP3	glibc 2.22	
Visual COBOL Development Hub 5.0 Enterprise Developer UNIX Components 5.0	AIX	7.1 TL4 SP3	IBM xIC.rte 13.1.3.1	C Runtime (xIC.rte package) is independent of the OS install, so Micro Focus has chosen to install these versions on our reference test machines.
		7.2 TL2 SP2	IBM xIC.rte 13.1.3.1	
	HP	11i v3 (2016 update)	aC++ runtime A. 06.28	C Runtime (aC++ package) is independent of the OS install. Micro Focus has chosen to install these versions on our reference test machines.
		11i v3 (2018 update)	aC++ runtime A. 06.25.01	
	Oracle Linux RH Kernel	7.6	glibc 2.17	
	Oracle Linux Unbreakable Kernel	6.10	glibc 2.12	
		7.6	glibc 2.17	
	RHEL	6	glibc 2.12	
		6.2	glibc 2.12	
		6.10	glibc 2.12	
		7.1	glibc 2.17	
		7.6	glibc 2.17	
	Solaris	11.4	Solaris Studio 12.6	C Runtime (Solaris Studio package) is independent of the OS install, so Micro Focus has chosen to install these versions

Product	Operating system	Version	C Runtime Version	Notes
				on our reference test machines.
	SUSE	12 SP2	glibc 2.22	
		15	glibc 2.26	

 **Note:** Enterprise Developer UNIX Components is an integrated component of the Enterprise Developer product, and was first introduced in Enterprise Developer 2.1.

Glossary of Definitions

Callable artifacts	Binary objects such as <code>.int</code> , <code>.gnt</code> , <code>.lbr</code> , <code>.dll</code> , and <code>.so</code> files that are loaded/called by a standalone executable.
Main executable	Binary objects that are executable by the system with a main entry point and are run as a separate process, for example <code>.exe</code> on Windows.
Object code	Binary output from the Compiler, created in platform-defined format (<code>.obj</code> , <code>.o</code>), that can be linked into a standalone executable or callable object, depending on the application.
Platform	The operating system and third-party components such as RDBMS clients and servers, Java App Servers, JRE and .NET framework, on which the COBOL application is built and executed.
Product Family	A set of product releases that have a common base. For example, Visual COBOL for Visual Studio 2012 and Enterprise Developer for Visual Studio 2012.
Relinking	The action of relinking links a main application executable with the latest run time system. Relinking does not require recompilation, but the process does require the original object files that were generated at the point of original compilation. Relink using the <code>cbllink</code> command line tool.