



Cumulative Changes from Net Express to Micro Focus Visual COBOL for Visual Studio

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

Copyright © Micro Focus 2018. All rights reserved.

MICRO FOCUS, the Micro Focus logo and Visual COBOL are trademarks or registered trademarks of Micro Focus IP Development Limited or its subsidiaries or affiliated companies in the United States, United Kingdom and other countries.

All other marks are the property of their respective owners.

2018-06-07

Contents

Cumulative Changes from Net Express to Visual COBOL for Visual Studio

.....	4
About this Guide	4
What was New	4
What was New in Visual COBOL 4.0	4
What was New in Visual COBOL 3.0	9
What was New in Visual COBOL 2.3 Update 2	15
What was New in Visual COBOL 2.3 Update 1	16
What was New in Visual COBOL 2.3	20
What was New in Visual COBOL 2.2 Update 2	27
What was New in Visual COBOL 2.2 Update 1	30
What was New in Visual COBOL 2.2	34
What was New in Visual COBOL 2.1 Update 1	40
What was New in Visual COBOL 2.1	42
What was New in Visual COBOL 2.0	44
What was New in Visual COBOL 2010	47
Significant Changes	64
Significant Changes in Visual COBOL 4.0	64
Significant Changes in Visual COBOL 3.0	69
Significant Changes in Visual COBOL 2.3 Update 2	70
Significant Changes in Visual COBOL 2.3 Update 1	71
Significant Changes in Visual COBOL 2.3	73
Significant Changes in Visual COBOL 2.2 Update 2	76
Significant Changes in Visual COBOL 2.2 Update 1	77
Significant changes in Visual COBOL 2.2	77
Significant Changes in Visual COBOL 2.1 Update 1	82
Significant Changes in Visual COBOL 2.1	82
Significant Changes in Visual COBOL 2.0	83
Unsupported or Deprecated Functionality	84
Unsupported or Deprecated at Visual COBOL 4.0	85
Unsupported or Deprecated at Visual COBOL 3.0	85
Unsupported or Deprecated at Visual COBOL 2.3 Update 2	85
Unsupported or Deprecated at Visual COBOL 2.3 Update 1	86
Unsupported or Deprecated at Visual COBOL 2010	86
Upgrading from Net Express to Visual COBOL	86
An introduction to the process of upgrading your COBOL applications	86
Compile at the Command Line Using Existing Build Scripts	88
Debugging Without a Project	89
Create a project and import source	89
Using Visual COBOL for Visual Studio	91
Change the Defaults to Replicate Your Existing Project Structure	93
Best Practice in Visual COBOL Development	94
Modernize Your Applications and Processes	95
Procedural COBOL Compared with Managed COBOL	96

Cumulative Changes from Net Express to Visual COBOL for Visual Studio

Welcome to Visual COBOL for Visual Studio. This document combines information on Visual COBOL for Visual Studio releases from the first release, Visual COBOL 2010 R1, to the most recent release. This information is taken from the various releases' Release Notes and other sources, and brought together here for your convenience.

You can use this document:

- If your Visual COBOL installation is up to date, to see the changes made in the latest release.
- If you already use Visual COBOL but your installation is not entirely up to date, to see the changes made over several updates, together in one place.
- If you are migrating from Net Express, to see everything that has changed in the life of Visual COBOL for Visual Studio.

About this Guide

We recommend that you read all the sections of this publication, looking for information on the release that is most relevant to your needs. The main sections are as follows:

What was New	This section describes new functionality that was introduced in each successive release.
Significant Changes	This section describes, for each successive release, the changes in behavior or usage that could affect the behavior of existing applications or impact the way the tools are used.
Unsupported or Deprecated Functionality	This section describes any functionality that was discontinued or deprecated at a given release.
Known Errors and Restrictions	This section describes known errors affecting the latest release, and restrictions in its use.
Upgrading from Net Express to Visual COBOL	This section gives guidance on upgrading from earlier Micro Focus products.

Within each section, we recommend reading in the reverse chronological order in which it is presented here - that is, reading about the most recent release first, and then going back through the development of the products. That way, you can more easily see if anything was added in an earlier release and subsequently removed.

What was New

This section describes the new features that were introduced in each successive release of Visual COBOL.

What was New in Visual COBOL 4.0

This release provides enhancements in the following areas:

- [Integration with Visual Studio](#)
- [Application Server JCA Support for Enterprise Server](#)

- [Build Tools for Windows](#)
- [Code Coverage](#)
- [Codeset support](#)
- [Compiler directives](#)
- [Data File Tools](#)
- [Database Connectors](#)
- [Docker](#)
- [Documentation on working with large applications](#)
- [Enterprise Server](#)
- [File Handler](#)
- [Library routines](#)
- [Micro Focus Unit Test Framework](#)
- [OpenESQL](#)
- [Platform support](#)

Integration with Visual Studio

[Back to Top](#)

This release includes the following editor improvements:

- Inline rename refactoring - overwriting a variable name directly in the editor now invokes rename refactoring so you no longer need to select **Refactor** from the editor context menu.
- **Extract to section** - a new quick action (Visual Studio 2015 and later only) is available in the editor. It enables you to refactor your code by adding more sections in the code. This can help you create more entry points that you can use during unit testing.
- Automatic insertion of END-EXEC, END-TRY and END-PERFORM statements - the closing statements are now automatically inserted after you have typed the opening statements.
- IntelliSense filters (Visual Studio 2017 and later only) - when IntelliSense displays a list of suggestions, a number of icons at the bottom of the IntelliSense window now enable you to filter the suggestions.

Working with files:

- Copybook graphs - a context menu command, **Show Copybook Graph**, in Solution Explorer enables you to display the graphical representation of the copybook dependencies of COBOL programs.
- **Open Folder** mode (Visual Studio 2017 only) - support is now available for editing, compiling and debugging files opened in Solution Explorer in Open Folder mode without having to create projects.
- The file property pages now display a list of all Compiler directives that apply to the file.

New IDE configuration option:

- You can use the **Limit IntelliSense Search Scope** setting in **Tools > Options > Text Editor > Micro Focus COBOL > Advanced** that helps improve the editor performance.

Relinking existing applications:

- You can now configure Visual COBOL to check whether applications created with older releases must be relinked. If the application uses an older version of the C runtime, Visual COBOL can automatically relink the existing executable or .dll to the new version of the C runtime without the need to recompile the application first.

If a project needs relinking, Visual Studio displays a message in the status bar (Visual Studio 2015 and 2017) or a dialog box (Visual Studio 2013) providing an option for you to choose and relink the project.

Application Server JCA Support for Enterprise Server

[Back to Top](#)

This release includes the following enhancements:

- COBOL Resource Adapters now support WebSphere 9.0 and WebLogic 12.2.1.
- Tomcat 7.0 support for servlet generation with J2SEBeans.
- NullSearch utility - for COBOL resource adapters, this new utility provides assistance in locating NULL fields in mappings passed to Enterprise Server. When a large number of arguments is provided in the parameters passed to Enterprise Server, it is difficult to locate NULL fields, which are not allowed. The NullSearch utility isolates NULL fields, so the Java application can be corrected.

Build Tools for Windows

[Back to Top](#)

This release includes Visual COBOL Build Tools for Windows, a separately-installable component of Visual COBOL that has been designed to be used in environments where you want to work with your COBOL projects but you don't want the overheads associated with the Visual Studio IDE.

Build Tools provide a lightweight, easy-to-install development environment that is well-suited for use in Docker containers and continuous integration or continuous delivery systems.

Code coverage

[Back to Top](#)

This release provides the following enhancements:

- In Visual Studio, the Micro Focus Code Coverage window now offers a File View in addition to the existing Program View.

Codeset support

[Back to Top](#)

Support has been added to enable codeset mapping to additionally be configured to use IBM's Conversion Tables directly instead of the Micro Focus supplied tables. You need to download IBM's conversion tables from IBM's Web site. Then you can use the MFCODESET environment variable to convert between IBM's CCSIDs.

Compiler directives

[Back to Top](#)

The following Compiler directives are new in this release:

- **DISPSIGN** - determines the display output of numeric fields with included signs, under an IBM mainframe dialect only.
- **GNTLITLINKSTD** - stops the suppression of call-convention 8 when both call-convention 2 and call-convention 8 are in effect for a `.gnt` file in an Intel x86 32-bit environment.
- **ILSMARTANNOTATE** - adds attributes to the items generated by ILSMARTLINKAGE, based on their data type, which can be used to identify the size or range of the item in COBOL.
- **ILSMARTTRIM** - trims any trailing spaces from a string item returned by the get property associated with an alphanumeric item processed by ILSMARTLINKAGE.
- **MAINFRAME-FLOATING-POINT** - specifies the format of a program's floating point data items: either IBM hexadecimal format or IEEE format. This directive is supported in managed code only.

The following Compiler directives contain new parameters in this release:

- **CHECKDIV** - a new parameter 'ACOS' now emulates a divide by zero operation on an ACOS mainframe system: the quotient and the remainder are set to the value of the dividend.
- **OCTRL** - a new parameter, `L`, specifies whether to include directory location comments in `.cls` and `.ins` inheritance files.

- **NUMPROC** - a new parameter 'ACOS' provides partial compatibility with the behavior of NEC ACOS COBOL processing of invalid data in USAGE DISPLAY data items and invalid sign information in USAGE COMP-3 data items.

Data File Tools

[Back to Top](#)

It is now possible to export any filtered results. When filtering a data file, you can use the results to create a new data file - click **Search > Export Results** when a filter is applied to save the filtered records to a new file. You can:

- Save the filtered records to a new file.
- Save the records that match the specified filter (such as customer information or orders).
- Download a subset of the data from a remote file.
- Save a small portion of the data for testing purposes.

Database Connectors

[Back to Top](#)

Debugging

[Back to Top](#)

This release includes the following enhancements:

- Enhanced .NET debugging in Visual Studio 2017 - a number of advanced debug features are now available when debugging .NET COBOL code. Features include support for performance tips, backwards debugging, an expression evaluator, integration with the Diagnostic Tools window, remote debugging of .NET COBOL applications, as well as support for IntelliSense in the debug windows (such as the Immediate, Watch, and QuickWatch windows).

Docker

[Back to Top](#)

This release provides support to enable you to run your COBOL applications in Docker containers, taking advantage of the many benefits offered by the Docker platform such as portability, performance, agility, isolation, and scalability.

Documentation on working with large applications

[Back to Top](#)

The product help now includes a new section, *Working with a Large Code Base*, that includes recommendations and best practices for working with large applications inside the IDE. It includes tips on how to structure your projects, how to optimize the performance of the IDEs, and step-by-step workflow showing how to move an existing legacy application into Visual COBOL.

Enterprise Server

[Back to Top](#)

The following enhancements have been made to Enterprise Server:

- Conversation filtering - the Enterprise Server Communications Process (MFCS) can now restrict access to listeners by client address. You can specify any permitted or forbidden addresses either by IP address, network mask, or domain name, and use wildcards. Filters can be applied to individual listeners, communications processes, or to entire regions. More specific filter rules override any general ones.

- The Enterprise Server Security Facility now starts throttling Verify requests when it receives more than 100 requests per second.
This can be used to limit the effectiveness of denial-of-service and brute force attacks. You can configure the value where throttling occurs. See *Verify Request Throttling* for more information.
- (Technology Preview only) Support for adding, deleting, and modifying XA resources in a live Enterprise Server region.
It is now possible to add, edit, or delete XA resources while an enterprise server instance is running. Any changes made come into effect after any in-flight transactions have completed. The ability to make these changes in a live environment comes under the control of the existing enterprise server permissions.
- XA-compliant Resources (XARs) - this release provides enhanced CTF tracing that allows more flexible reporting of warnings and errors on the RM switch module level.
- A new Communications Server resource class - enables you to control the access to the Enterprise Server Console Log and Communications Server Log when external security is in effect for an enterprise server region; see *Resource Classes for Communications Server* for more information.
- Improved catalog availability - there is now an improved resilience to temporary communication issues with the catalog and error reporting enabling a region to stay active if a region has multiple catalogs defined and one of the catalogs is not available.
- Enhanced SSL/TLS certificate support - for communications with TLS (formerly SSL), additional certificate and key file formats are supported. Servers may now be configured with both an RSA and an ECC key and certificate.
- Enhanced SSL/TLS cipher configuration - for communications with TLS (formerly SSL), the permitted cipher suites and their preferred order can now be configured. The minimum size of Diffie-Hellman groups for DH key exchange can also be configured. The defaults have been made more secure.

File Handler

[Back to Top](#)

This release provides the following enhancements:

- The DFSORT and SYNCSORT emulations now support the NULLOFL parameter of the OUTFIL statement.
- The **ASCIISOSI** configuration option is now available. It adds the required SOSI characters to the relevant EBCDIC DBCS character strings in order for them to be displayed or written out correctly.

Library routines

[Back to Top](#)

The following library routines contain new functionality:

- **CBL_GET_OS_INFO** - this library routine can now detect if the program is running within a Docker container: `cblte-osi-rts-capabilities` parameter, bit 7.

Micro Focus Unit Test Framework

[Back to Top](#)

This release provides support for the following functionality:

- Generation of unit test stubs for selected entry points within your program.
- Support has been added to Visual Studio for unit testing of managed procedural projects.

OpenESQL

[Back to Top](#)

This release provides the following new features:

- Support for SQL Server 2017.
- The SQL(TRANSACTION) compiler directive has been enhanced to clearly define transaction boundaries.
- A new SQL(NOWHERECURRENT) compiler directive that allows you to define updateable cursors that do not do positioned updates or deletes with PostgreSQL or MySQL.
- Larger communication area (PID) that accommodates longer plan and program names.
- SQL(OPTIMIZECURSORS) has been enhanced for consistent and better cursor performance across all OpenESQL backends.

Platform support

[Back to Top](#)

Note the following changes in platform support for this release:

- Windows 8 and Windows Server 2012 are no longer supported for developing applications. They are still supported for deployment.

What was New in Visual COBOL 3.0

Visual COBOL 3.0 provided enhancements in the following areas:

- [Integration with Visual Studio](#)
- [Application Server JCA support for Enterprise Server](#)
- [Azure support](#)
- [Building applications](#)
- [COBOL language enhancements](#)
- [Code analysis](#)
- [Code coverage](#)
- [Compiler control](#)
- [Data File Tools](#)
- [Database access - DB2](#)
- [Database access - MySQL](#)
- [Database access - OpenESQL](#)
- [Documentation](#)
- [Enterprise Server](#)
- [Interface Mapping Toolkit](#)
- [iFileshare](#)
- [Micro Focus Unit Testing Framework](#)
- [XML processing](#)

Integration with Visual Studio

Visual COBOL 3.0 provided enhancements in the following areas:

- Integration with Microsoft's Visual Studio 2017, the most recent version of the world's most popular integrated development environment (IDE) for the Windows platform, offering significant benefits for developers and businesses developing software for Windows.
- COBOL Editor:
 - Brace completion - a closing quote mark or bracket is automatically inserted when you type the opening quote mark or bracket at the end of a line.

- Brace matching for IF and EVALUATE statements - clicking on any of the clauses of these statements highlights the opening and closing statements and all clauses. Use **CTRL +]** to move the cursor to the next clause in the statement.
- Collapse to definitions - closes all sections, entry points and methods in an open file. Use the **Outlining > Collapse to Definitions** editor context menu command.
- Colorization is now available for QuickInfo, completion tooltips and signature help.
- Completion of statements - IF and EVALUATE statements are now automatically aligned and completed.
- Indentation - smart indentation is now available to control how lines indent within IF and PERFORM statements and after continuation statements.
- Locate definition, , on the COBOL toolbar enables you to search for data items and identifiers by specifying any string of characters that might be part of their name. This command replaces **Edit > Go To Location**.
- Outlining - outlining is now available for 01 level group items (for expanding and collapsing the entire group)
- Quick actions (a feature of Visual Studio versions 2015 and later) - Visual COBOL now supports Visual Studio's quick actions for rename refactoring, and finding or creating missing copybooks.
- Tooltips are now available for managed COBOL members.
- COBOL toolbar - the commands on the COBOL toolbar have been updated in order to optimize your editing experience. A number of commands (renumber and unnumber) have been removed from the context menu in the editor and are now only available from the toolbar.
- Directives determination - there is now a preview dialog box when the IDE scans your sources for Compiler directives. This enables you to review and approve the directives before they are set.
- Error reporting - you can now sort the errors and warnings in the **Error List** window by their COBOL error code.
- Import, export and synchronization of the IDE options - you can now export a number of the COBOL global options as a Visual Studio .settings file. You can import the file into other instances of Visual COBOL.

Visual COBOL for Visual Studio 2015 and later now supports the automatic synchronization of a number of COBOL settings across the copies of Visual Studio installed on different machines. The settings are synchronized through your MSDN account. See the MSDN for more details and for information on any restrictions to the functionality.

- **Project Details** window - provides a number of enhancements, including performance improvement. It is now possible to filter on project or solution, perform a search, and sync with Solution Explorer. Tooltips are available in the Overrides column to show the override directives.
- Net Express import wizard - provides a new option for including copybooks in a project.
- Rename refactoring - this release introduces rename refactoring for COBOL code that can help to improve the readability of COBOL elements or make their purpose clearer. You can rename elements such as variables and identifiers, section and paragraph names, classes, and methods across a program or a solution.
- Support for standalone files - various improvements have been made to how you work with files that are not part of a project:
 - When you first open a standalone file, the editor displays an alert bar with options to browse for a symbols (.idy) file to use to compile the standalone file or to configure the Compiler directives.
 - You can now use the file's properties window to edit the Compiler directives and to specify options for debugging.

In previous versions of Visual COBOL, standalone files were known as "single files". References to "single files" in the IDE and the product help have been changed to "standalone files".

Application Server JCA support for Enterprise Server



Restriction: This feature applies only when the Enterprise Server feature is enabled.

In Visual COBOL 3.0, EJBGEN has been updated to generate an EAR file as a part of the COBOL deployment process, which enables you to deploy EJBs to Java Application Server.

Azure support

Visual COBOL now supports the following versions of Microsoft Azure SDK - version 2.9.6 with Visual Studio 2013 and version 3.0 with Visual Studio 2015 and later.

Building applications

Visual COBOL 3.0 provided the following improvements:

- Support for faster, parallel building on multi-CPU machines - support has been added for multi-processor compilation of the sources in native COBOL projects on multi-CPU machines.

You can specify the maximum number of concurrent compilations from the IDE preferences - **Tools > Options > Micro Focus > Projects**. In Visual Studio, multi-processor compilation must also be enabled in the project's properties.

COBOL language enhancements

Visual COBOL 3.0 includes the following enhancements to the COBOL syntax:

- The DISPLAY-OF and NATIONAL-OF intrinsic functions are now able to process conversions using any IBM CCSID value.

The following enhancements are available in managed COBOL:

- To avoid an exception being thrown if an explicit conversion fails, use the AS IF syntax, which results in the target object being set to null and no exception thrown.

Code analysis

Visual COBOL 3.0 provided the following improvements:

- A new group of predefined rule sets for 64-bit readiness is now included in Visual COBOL.
- Support for importing code analysis reports produced with one of Micro Focus's advanced tools for code analysis, Enterprise Analyzer or COBOL Analyzer.

Code coverage

The following improvements are available within the IDE:

- Information about unexecuted programs - the code coverage reports in the **Code Coverage** window now show the unexecuted programs.
- Code coverage support for standalone COBOL files - you can import existing code coverage reports in the **Code Coverage** window and use it to supply code coverage information for standalone files.

If you are using Test Coverage from the command line, you can now use the following features:

- A new Compiler directive, COLLECTION - the directive enables test coverage to gather information about unexecuted programs. In the IDE, this directive is automatically set on a project when you enable code coverage for it.
- A new command line utility, tcutil - the utility enables you to convert the test coverage binary results file into XML format.
- It is now possible to integrate test coverage in a Continuous Integration (CI) system. You can use tcutil and an XSLT processor to transform test coverage data into a format suitable for including in a CI.

Compiler control

The following Compiler directives are new in this release:

- **COLLECTION** - provides a mechanism for code coverage to identify unexecuted programs.
- **ILSTDLIB** - helps you to ensure that your .NET COBOL code compiles with the correct version of the .NET Framework Microsoft Common Object assembly `mscorlib.dll`. When building from Visual Studio or using MSBuild to compile a COBOL project, the `NOILSTDLIB` directive is set, and an `ILREF` is generated pointing at the version of `mscorlib` appropriate for the target framework selected in the project's properties.

The following Compiler directives have been updated:

- **ALIGN** - this directive has new parameters (`FIXED` and `OPT`) that can be used in conjunction with the integer taken, which can aid performance. The default is `ALIGN"8 OPT"`; see the Comments section of the *ALIGN* Compiler directive topic for details of its affect on memory boundaries.
- **ARITH** - this directive emulates the IBM mainframe option of the same name. Defines the maximum number of digits for numeric data items.
- **FASTINIT** - this directive is now on by default when setting the MF dialect; it remains not set by default for other dialects.
- **SSRANGE** - this directive now has an additional option (`3`), which permits zero-length reference modified items at run time when bounds checking.
- **XMLPARSE** - includes a change in the way entities are processed when `XMLPARSE"COMPAT"` is set

Data File Tools

The Data File Tools editor previously provided (from Visual COBOL 2.3) as a Technology Preview item was supported at GA level from Visual COBOL 3.0.

Visual COBOL 3.0 provided the following enhancements to Data File Tools:

- **Opening files in shared mode** - it is now possible to switch between read-only shared and edit modes. While a file is open in shared mode, others users can only open it in shared mode to ensure data consistency between users.
- **Enterprise Server-level of security when accessing files** - there is an improved level of security when exchanging data between Data File Tools and the targeted enterprise server instance. Users must now provide a user ID, group and a password when they try to access and view datasets in enterprise server instances. These are used for authentication and authorization checks to provide the same access level as Enterprise Server.
- **Opening datasets using SSL** - communication to a region is now possible using SSL. To enable the SSL communication, you need to provide a Java trust store which contains either a CA root certificate or a self-signed certificate of the region that it is communicating to. Java and the targeted region SSL configurations need to meet each other's standards in order for the communication to succeed.

This feature enables you to secure the information exchange between Data File Tools and the targeted enterprise server.

- **Auditing of access and updates on datasets** - Audit Manager now audits the access and updates on datasets via Data File Tools.
- **Support for existing .pro files** - enables you to use your existing editor profiles.
- **Support for existing .str files** - enables you to use your existing COBOL structure files.
- **Automatic timeout** - if no internal operations or external actions (such as a mouse click) have been detected for 30 minutes, Data File Tools now displays a countdown message. If the user does not take any decision within the specified period, Data File Tools closes all opened files.

Database access - DB2

Visual COBOL 3.0 provided a new `DB2"QUALIFY-CALL"` Compiler directive that enables stored procedure invocations to include a schema name.

Database access - MySQL

Visual COBOL 3.0 provided support for MySQL with ODBC.

Database access - OpenESQL

Visual COBOL 3.0 provided the following new features:

- Statement prefixes for the SQL "CHECK" Compiler directive that enable the creation of temporary tables and other SQL objects at compile time, ensuring full SQL syntax checking during compilation.
- SQL "OPTIMIZECURSORS" Compiler directive that enhances processing for traditional embedded SQL cursors that use WITH HOLD and FOR UPDATE clauses.
- SQL "CLOSE_ON_COMMIT" Compiler directive to leave cursors open for further result set processing after a commit.
- SQL "GEN-SQLCA" Compiler directive that generates an SQLCA similar to the z/OS DB2 directive STDSQL "YES".

Documentation

The following new section have been added to the product help:

- *Where do I start?* - located on the launch page of the product help, this section provides the information you need in order to get started depending on which aspects of the product you need to get to grips with first.

Enterprise Server

Improvements are available in the following areas:

Integration with Visual Studio

- Exporting an enterprise server definition from the IDE in XML format.
- Importing an enterprise server into the IDE using its definition file.
- In Visual Studio, it is now possible to configure an enterprise server to produce a core dump file directly from the IDE, from the server properties.

Long user IDs and passwords:

- Enterprise Server now supports user IDs and passwords of up to 100 characters. It is possible to map IDs from long to short (or vice versa) to enable compatibility with programs that do not support long names.

SHA-256 support in DemoCA:

- By default, the Demonstration Certificate Authority (CA) now signs certificates with SHA-256. This ensures that the demonstration or evaluation certificates will be accepted by modern browsers and other software that has enhanced security requirements.

Syslog auditing:

- Enterprise Server now supports auditing using syslog events, which can be consumed by a wide range of Security Information and Event Management (SIEM) products. This replaces the Audit Manager auditing solution. Syslog auditing provides a much more efficient auditing mechanism, with significantly less impact on overall speed.

Interface Mapping Toolkit

The Interface Mapping Toolkit (IMTK) contains the following enhancements in Visual COBOL 3.0:

- Cross Origin Resource Sharing (CORS) support for REST Web services, enabling Web service access from a CORS-supported Web browser.

iFileshare

iFileshare, previously considered a Technology Preview feature, was supported at GA level from Visual COBOL 3.0. It also contained the following enhancements:

- An improved failover and recovery process. iFileshare now supports full recovery of nodes in the group. For high availability (HA-VSAM) groups, servers can now rejoin the group without the entire group having to be restarted. In addition:
 - A primary failover now results in a takeover from the most suitable node.
 - If configured, external clients will automatically reconnect to the new primary and will issue a notification if the transaction has been lost.
 - A failed node, when restarted, will rejoin the group, recover its files and request a log update from the current primary. Once this task has completed it will be considered an active hot-standby and will continue to process replication requests as normal.
 - Users will experience a higher level of uptime/availability with their Fileshare configuration and will be able to recover from errors more easily.
- A new exit procedure, `ifsexitproc.cbl`, can be configured to automate some aspects of iFileshare behavior.
- The iFileshare Control page in ESMAC contains details of the current iFileshare high availability group.
- The following new iFileshare-specific environment variables are available:
 - `FSWRKDIR` - enables you to specify the Fileshare working directory, overriding the default, which is the system directory of the region.
 - `FSCHKLFH` - determines if a check is performed when a high availability group is started, to test the consistency of the data files within the group.
- The database reference file (`dbase.ref`) now supports wildcard matching for filenames, allowing you to perform operations on multiple files at once; for example: `fs /d dbase.ref /f data*` adds the entire contents of the data directory to the database reference file.

The Micro Focus Unit Testing Framework

The Micro Focus Unit Testing Framework is now available from within the IDE. It includes much of the architecture you would expect of an xUnit framework to create, compile, run and debug unit tests, including the following features:

- A unit test project template.
- A test creation wizard that enables you to generate tests from your source code.
- Code snippets for each element of a test case.
- Support for running tests with Code Coverage enabled.
- The Micro Focus Unit Testing window, where you can manage your test runs and view test output.

There has been a number of enhancements to the command line version of the Micro Focus Unit Testing Framework. Support has been added for:

- Generating NUnit-style reports.
- Running test fixture files using Apache Ant.
- Applying traits to your test cases, then performing a test run based on those traits.
- Applying a high, medium, or low priority to test cases, which affects the order in which they are run.
- Adding coded command line options directly into your test code.
- Using a test run-specific configuration file, in which you can set environment variables.

XML processing

XML PARSE now works in a purely managed COBOL environment. It is now supported in JVM COBOL and, in both .NET and JVM COBOL, it has a fully managed implementation. XML PARSE working without calling out to native code ensures it can be used in restricted rights environments.

What was New in Visual COBOL 2.3 Update 2

Visual COBOL 2.3 Update 2 provided enhancements in the following areas:

- [Integration with the Visual Studio IDE](#)
- [Building applications](#)
- [COBOL language enhancements](#)
- [Classic Data File Tools](#)
- [Compiler directives](#)
- [Debugging applications](#)
- [Dialog System GUI](#)
- [File handling](#)
- [Library routines](#)

Integration with the Visual Studio IDE

Visual COBOL 2.3 update 2 provided the following enhancements in the integration of Micro Focus COBOL with the Visual Studio IDE:

- A new menu command **SQL Option for DB2** on the **Tools** menu enables you to access all the SQL Option for DB2 utilities.
- It is now possible to specify options for compiling native resource files (.rc) by accessing the file's properties from within the IDE.
- Editor improvements:
 - Colorization of the code in the ToolTips for collapsed regions in the COBOL editor has been added.
 - The Time and Date Warp options are now available on the **Run-Time Configuration** tab when editing an application configuration file for native COBOL projects.
 - **Go To Next Method** and **Go To Previous Method** editor context menu commands are now available for native Object-Oriented and for managed COBOL code.
 - Improvements have been made to editing performance.

Building applications

It is now possible to use the 64-bit version of MSBuild to build COBOL projects.

COBOL language enhancements

Numeric, edited and external floating point items can now specify USAGE NATIONAL when the NATIONAL"2" Compiler directive is in effect. Signed numeric items must be specified with the SIGN IS SEPARATE clause.

Classic Data File Tools

A new command line utility is available which enables you to initiate the following actions: open data files, create or open record layout files, create or open segment layout files, and open IMS databases using a DBD or PSB file. Note that although you can initiate these actions from the command line, you must complete them from within the IDE.

Compiler directives

The following Compiler directives are new in this release:

- **COMMAND-LINE-LINKAGE** - enables you to call a program and pass the command line to the main program as a parameter to be accessed via the Linkage Section. This offers equivalent functionality to the `command_line_linkage` tunable, which has now been deprecated.

- EBC-COL-SEQ - controls the behavior of an EBCDIC collating sequence, specified in a NATIVE"EBCDIC" program. EBC-COL-SEQ"1" (the default) maintains use of the long-standing fixed (platform-independent) EBCDIC collating sequence. EBC-COL-SEQ"2" prompts use of the latest CODESET table, which varies according to platform and user-controlled MFCODESET environment variable setting.
- NATIONAL - enables you to specify numeric, edited and external floating point items as USAGE NATIONAL.

Debugging applications

- You can configure the debug tooltip for subscripted OCCURS items to either display the elements of arrays or to display the value of the current expression. To do this, right-click in the editor, click **COBOL Debug Tooltip Style** and either enable or disable **Show COBOL OCCURS or reference modification value**.

Dialog System GUI

Installing the version of the Compatibility AddPack for Visual COBOL provided with this release adds a new menu item, **Micro Focus Dialog System**, to the **Tools** menu of Visual Studio for starting the Dialog System GUI from within the IDE.

File handling

MFJSORT ICETOOL now supports the USING parameter in the SELECT operator.

Library routines

The following library routine contains new functionality:

- CBL_GET_PROGRAM_INFO - a new function (function 10) has been added for native COBOL which returns the path and program name, or the program name only of a particular program.

What was New in Visual COBOL 2.3 Update 1

Visual COBOL 2.3 Update 1 provided enhancements in the following areas:

- [Integration with the Visual Studio IDE](#)
- [Application Server JCA support for Enterprise Server](#)
- [Code Analysis](#)
- [Code Coverage](#)
- [Compiler directives](#)
- [Database Access - OpenESQL](#)
- [Data File Tools](#)
- [Debugging applications](#)
- [Dialog System Applications](#)
- [Editor writing assistance](#)
- [Enterprise COBOL 5.2](#)
- [File Handling](#)
- [Library routines](#)
- [Managed COBOL Syntax](#)
- [Native COBOL Syntax](#)
- [RM/COBOL compatibility](#)
- [Rosetta Stone for COBOL, .NET and Java Developers](#)

Integration with the Visual Studio IDE

Visual COBOL 2.3 Update 1 provides the following enhancements in the integration of Micro Focus COBOL with the Visual Studio IDE:

- The **Go To location** and **Go To Procedure Division** commands are now available from the **Edit** menu. In addition, the **Go To Procedure Division** has a shortcut key - **Ctrl+K, Ctrl+J**.
- Object Browser - now provides support for .NET COBOL code. You can view the members of objects in your project and the definition of classes and methods as specified by the XML documentation comments in your code. The window also provides support for the **Go To Definition** command.
- A new search option, **Current COBOL Program**, has been added to the **Find in Files** dialog box. This enables you to search in the COBOL program currently opened in the editor and in any copybooks referenced by that program.
- Solution Explorer - a new context menu command, **Add Folder to Copybook Paths**, is now available for the subfolders of your projects. This enables you to add the subfolders to the copybook path of the project.

Editor improvements:

- Line numbers - a new option for configuring the increment for the COBOL and the standard line numbers when you use Renumber and Unnumber has been added. You can specify the increment from **Tools > Options > Text Editor > Micro Focus COBOL > Line Numbering**.
- Outlining - a new IDE option enables you to switch outlining in the COBOL editor on or off. See the **Enable Outlining** option available on the **Advanced** page in **Tools > Options > Text Editor > Micro Focus COBOL**.
- Program breakpoints - a new context menu command, **Add COBOL Program Breakpoint**, enables you to set program breakpoints directly from the editor.

Code analysis

[Back to Top](#)

This release provides support for performing code analysis at the command line using Microsoft's MSBuild utility which enables the integration of code analysis in CI frameworks.

Support is available for various MSBuild parameters. You can use MSBuild with the /p switch and with the following Micro Focus-specific command line options:

- `RunMicroFocusCodeAnalysisAfterBuild` - controls whether code analysis is performed with MSBuild or not regardless of what code analysis options are specified the project's properties.
- `ActiveRulesets` - specifies which rule sets should be used when running code analysis from the command line.

Code coverage

[Back to Top](#)

The code coverage reports are now integrated with the IDE and with the editor. Features include:

- A new Code Coverage window showing the statistics of what percentage of the code has executed.
- Navigation from the Code Coverage window to the missed and covered blocks in the editor.
- Colorization in the editor of blocks that were executed (covered blocks) or not (missed blocks).

Compiler directives

The following Compiler directives are new in this release:

- `ILMAIN` - you now specify the main entry point for the executable program, which can be specified either as `class-name::method-name`, or just as `method-name`. For example, `ILMAIN"classA::methodB"` or `ILMAIN"methodB"`. The first format can be used to distinguish between multiple methods with the same name in different classes.

- OOCTRL - a new parameter, +/-A, as been added. Set this parameter to -A to allow ActiveX controls in your COBOL application to use classes and methods in the OLE class library. The default is +A, which does not allow it

Database Access - OpenESQL

[Back to Top](#)

This release provides the following new features:

- Demonstrations for using OpenESQL in .NET applications. To view these demos, see the *SQL* demonstrations in the Visual COBOL Samples browser.
- Support for the Oracle fully managed ODP.NET driver.

Data File Tools

This release provided improved security and increased support for more file types. Features include:

- Certain aspects of Enterprise Server security are honored when you attempt to access data sets. If the Enterprise Server region has security enabled, logon details must be authenticated before you can access the data set. If the details are unable to be authenticated, access is denied.
- When using a record layout, certain data is now validated at field level (to ensure the contents is compatible with its picture string) and record level (to ensure the record length matches the layout size).
- Full editing support has been added for variable block sequential files and relative files. Full editing is also available for line sequential files, as long as they do not contain any binary data

Debugging applications

- You can configure the debug tooltip for subscripted OCCURS items to either display the elements of arrays or to display the value of the current expression. To do this, right-click in the editor, click **COBOL Debug Tooltip Style** and either enable or disable **Show COBOL OCCURS or reference modification value**.

Dialog System Applications

[Back to Top](#)

The version of the Compatibility AddPack for Visual COBOL released with Visual COBOL 2.3 Update 1 enables you to give your Dialog System applications a refreshed user interface. Features include:

- Support for a modern look and feel for Dialog System applications.

Dialog System now supports Microsoft's visual styles for controls and fonts to give your existing Dialog System applications a modern look that is native to the Windows version the applications are running on.

You use a new environment variable, MFVSSW, to switch the visual styles on. Alternatively, for applications that compile to an executable, to switch the new styles, you can add an application manifest file to the applications' project files.

See the section *Modernizing Dialog System Applications* in your product help for details on how to enable the visual styles, and for information about any possible changes in behavior or appearance of the common controls.

- A modern look and feel of the Dialog System painter.

There are new configuration options in the Dialog System painter for applying the visual styles and fonts to the painter.

- Visual Studio project templates for creating Dialog System applications with a modern look out-of-the box.

The Compatibility AddPack for Visual COBOL now installs the following new project templates:

- **Dialog System Application (Modern)** for creating Dialog System applications that use Microsoft's visual styles by default.
- **Dialog System Application (Classic)** for creating Dialog System applications that use the classic fonts and look.
- Additional samples:

All of the Dialog System samples previously available with Net Express are now included with the Compatibility AddPack for Visual COBOL. The samples have been converted to Visual Studio solutions.

For more information see the section *Modernizing Dialog System Applications* in your product help and the Release Notes for the Compatibility AddPack for Visual COBOL.

Enterprise COBOL 5.2

[Back to Top](#)

With the introduction of Enterprise COBOL 5.2, the following features were supported:

- The VOLATILE keyword is supported within the data entry description; although, this is treated as documentary. It has also become a reserved word when under the ENTCOBOL dialect.
- Format 2 of the SORT statement no longer treats the COLLATING SEQUENCE clause as documentary-only.
- The SUPPRESS clause of the XML GENERATE statement has been enhanced.
- The IBM z/OS JSON parser API, as documented for the IBM z/OS client web enablement toolkit.

File Handling

- A new indexed file format, IDXFORMAT12, has been introduced to improve file maintenance and recovery procedures when using the rebuild utility. This file format is similar in structure and use to IDXFORMAT8. Where the two formats differ is that an IDXFORMAT12 file has an accompanying side file (.idx file) containing the indexed key information.

You can use this type of file with the new `rebuild /q` option. This rebuild process is considerably quicker than other rebuild processes such as a data scrape or `rebuild /p`.

- Faster SORT operations for fixed block records - when using the DFSORT emulation, the performance when sorting fixed block records has greatly improved.

Library routines

The following library routine were new at this release:

- CBL_CODESET_SET_MAPPING - enables you to change the codeset in effect.
- CBL_RUNTIME_ERROR - forces an application to terminate with a run-time error condition.
- PC_PRINTER_INFO_DOTNET (.NET COBOL only) - enables the access to the native Hdc for a printer.

Managed COBOL syntax

[Back to Top](#)

The following enhancements have been made to the managed COBOL syntax:

- You can now create generic iterators.
- You can now use the Profiler utility to obtain detailed statistics on the run-time performance of managed COBOL applications.

Native COBOL Syntax

[Back to Top](#)

The following items are new features of the native COBOL syntax:

Class condition tests New and updated class condition tests are available for DBCS, KANJI, and JAPANESE.

RM/COBOL compatibility

[Back to Top](#)

The RM/Panels syntax is now supported in Micro Focus COBOL applications.

Rosetta Stone for COBOL, .NET and Java Developers

[Back to Top](#)

The product Help now includes a quick and easy to use syntax guide for developers who need to learn OO COBOL syntax when modernizing COBOL applications for the Java or .NET platforms. The guide includes side-by-side equivalent syntax for COBOL, C#, VB and Java.

Windows Azure

[Back to Top](#)

Visual COBOL support on Microsoft Azure has been updated to version 2.8 of the Microsoft Azure SDK.



Important:

- Starting with this release, versions of the Microsoft Azure SDK earlier than version 2.8 are no longer supported.
- Support for the Microsoft Azure SDK is no longer provided in Visual COBOL for Visual Studio 2012. If you have COBOL Azure projects created with earlier versions of Visual COBOL for Visual Studio 2012, to maintain them, use Visual COBOL for Visual Studio 2013 or 2015.

What was New in Visual COBOL 2.3

Visual COBOL 2.3 provided enhancements in the following areas:

- [Integration with Visual Studio 2015](#)
- [General IDE enhancements](#)
- [Building applications in Visual Studio](#)
- [COBOL editor in Visual Studio](#)
- [Code analysis](#)
- [Code coverage](#)
- [Command Line Compilation and Linkage](#)
- [Compiler directives](#)
- [Converting additional directives to project's properties](#)
- [Data File Structure command line utility](#)
- [Data File Tools \(Technology Preview\)](#)
- [Database access](#)
- [File handling](#)
- [Library routines](#)
- [Managed COBOL syntax](#)
- [Micro Focus Infocenter](#)
- [Micro Focus Unit Testing Framework](#)
- [Microsoft Azure](#)
- [Personal edition licensing](#)
- [Preprocessors](#)
- [Profiler](#)

- [REST service interfaces](#)
- [RM/COBOL Compatibility](#)
- [Single file support](#)
- [Tunables](#)
- [Updated run-time system](#)

Integration with Visual Studio 2015

This release provides a new Visual COBOL flavor that takes advantage of the new features of Visual Studio 2015, including:

Light Bulbs	You can use the light bulb feature () to quickly implement unimplemented interfaces or to resolve unknown types.
Compatibility (Project Round-Tripping)	Visual COBOL supports the Visual Studio project round-tripping feature which enables you to work with COBOL projects in Visual Studio 2012, Visual Studio 2013 or in Visual Studio 2015 without the need to upgrade the project file.
.NET Framework 4.6 Compatibility	Provides support for creating managed COBOL applications that target version 4.6 of the .NET Framework.
Microsoft Help Viewer 2.2	Provides support for the Micro Focus Help in Microsoft Help Viewer 2.2.

General IDE enhancements

In Visual Studio:

- Menu items - the following Micro Focus utilities were previously available under the **Tools > Micro Focus** menu and have been moved as follows:
 - **ADO.NET Connection Editor** - now accessed from **View > Micro Focus SQL Tools**.
 - **OpenESQL Assistant** - now accessed from **View > Micro Focus SQL Tools**.
 - **OpenESQL Configuration Utility** - now accessed from **View > Micro Focus SQL Tools**.
 - **Web Service Client** - now accessed by creating a new project of type "Web Service Client Application from JSON/REST" - click **File > New > Project > COBOL > Native > Web Service Client Application from JSON/REST**.
- Managed COBOL project properties - a new setting, **Expose group linkage items to managed code**, is now available on the **COBOL** tab in the properties of managed COBOL projects. Checking this sets the ILSMARTLINKAGE directive that specifies that group linkage items in your code are to be exposed as properties of a class. A related button, **Options**, enables you to also specify whether to remove specific prefixes from the names of COBOL data items, whether to make the new classes serializable, expose Linkage Section items as nested classes of the program class or to limit the property generation to non-redefining elementary items.
- Resetting the file properties - a new button, **Use Project Defaults**, on the file property pages of COBOL files in native projects now enables you to reset the file properties of COBOL files in native COBOL projects.

Building applications

Visual COBOL now supports Visual Studio's parallel builds for COBOL projects. Parallel building enables you to build multiple projects faster on multi-CPU machines. Following this change, in order for your multi-project solutions to build in parallel successfully, ensure that the project dependencies and build order are set correctly for your solution using **Project > Project Dependencies**. For more information about parallel building, see *Tips on Building COBOL* in your product help and your product Release Notes.



Note: Parallel builds are not supported with Personal Edition licensing.

COBOL editor

The COBOL editor includes the following enhancements and new features:

- Improved Intellisense writing assistance for COBOL in both native and managed COBOL:
 - Context sensitive suggestions - IntelliSense only shows suggestions that are relevant for the position of the cursor in the code or for the type of project.
 - Enhanced completion lists - lists include any relevant COBOL verbs, clauses and words, copybooks, code templates, data items and section and paragraph names.
 - Intelligent assistance with completing statements - when you have entered a COBOL verb, IntelliSense shows suggestions for the relevant clauses and identifiers that you can use to complete the statement.
 - Automatic completion for single items - IntelliSense automatically inserts single suggestions in the code.
 - Qualifying non-unique names - IntelliSense qualifies data items whose names are not unique.
 - Configuration preferences for IntelliSense - enable you to configure what suggestions appear in the completion lists, whether suggestions are added in insert or overwrite mode, and the case of the inserted words.
 - Snippets - code snippets are now included in the IntelliSense suggestions.
- \$IF-\$ELSE-\$END statement colorization - parts of a \$IF Compiler-control statement that are not executed are now colored in grey.
- \$REGION statement - support is provided for the \$REGION Compiler-control statement. You can use \$REGION - \$END-REGION to surround blocks of code that you want to fold or expand in the editor.
- Copybook glyphs (📄) in the left-hand margin on a copy line - indicate that a copybook can be expanded inline.
- Creating copybooks from selection of code - a new context menu command, **Extract to Copybook** in the editor enables you to move a selection of COBOL code into a new copybook file. The file is added to the project and the code in the original program is replaced with a COPY statement that refers the new copybook.
- Expanded copybook view - provides indicators for read-only copybooks
- Outlining - outlining is now available on comment blocks, paragraphs and on \$REGION and \$IF statements.
- Task List comments - the COBOL editor now provides support for Visual Studio-style Task List comments in COBOL. Typing TODO, HACK or UNDONE in the code immediately after the declaration of the COBOL comment (*>, *>> or * in column 7) creates a task that appears in the **Task List** window.

Code analysis

Visual COBOL now offers more advanced code analysis features and enables you to run various analysis queries (rules and groups of rules called rule sets) against your code to ensure adherence to standards such as standards for coding or performance.

You can run analysis rules against programs in a project in the IDE at user request or you can run analysis rules at the end of a project's build.

Code coverage

Visual COBOL now provides support for code coverage of native COBOL applications directly from within the IDE where code coverage uses the Test Coverage functionality. You can produce code coverage reports for applications running in the COBOL run-time and for applications that run in Enterprise Server.

To produce reports, you need to enable code coverage in a project's properties, compile your application and then run your application with code coverage to produce the relevant reports. For applications that require an Enterprise Server instance, you start the enterprise server with code coverage.

Command Line Compilation and Linkage

When using the `cbllink` command to compile and link, there is a new `-y` option. Use this option to create an executable that includes support to be able to run on Windows XP and Windows Server 2003.

Compiler directives

The following Compiler directives are new in this release:

EOF-1A	Treats a 0x1a character in the source file as the end of file.
NLS-CURRENCY-LENGTH	Specifies the number of bytes to allocate for the currency symbol in a PIC field.
NULL-ESCAPE	Treats a 0x00 character in the source file as an escape character for other non-printable characters in the source code.

The following Compiler directives contain new parameters in this release:

DBSPACE The new parameter 'MIXED' extends the DBSPACE directive to be able to evaluate data items in programs that contain a mix of single-byte and double-byte strings.

Converting additional directives to project's properties

The **Update Project Properties** utility allows you to convert additional directives in your project to project properties. This option is presented on opening a project. You can select to not show this dialog again by unchecking **Check Additional Directives for project properties** in **Tools > Options > Micro Focus > General**.

Data File Structure command line utility

The Data File Structure Command Line (DFSTRCL) utility is a DOS-based command line utility that enables you to create record layout (`.str`) files from COBOL debug information (`.idy`) files. You can use the utility to process a single `.idy` file or batch process up to 100 `.idy` files.

Data File Tools (Technology Preview)



Note: This is a technology preview feature only. It is being made available to allow you to test and provide feedback on this new capability; however, this feature is not intended for production use and it is not supported as such. Furthermore, Micro Focus does not guarantee that this feature will be delivered at a GA level and if it is, then the functionality provided might differ considerably from this technology preview.

The Data File Tools (Technology Preview) is a new standalone text editor in which you can create and edit data files. By nature of it being a 'technology preview' product, it does not currently include all the functionality that was available in the previous version of Data File Tools - now referred to as Classic Data File Tools. If you require any of the functionality not provided in this version, you can still use the classic version by accessing it in the usual way.

To run Data File Tools (Technology Preview), type `mfdatatools2` from Visual COBOL's command prompt or a terminal.

To use the new editor directly from the Visual Studio IDE, clear the **Use classic data file tools** check box, available from **Tools > Options > Micro Focus > Data File Tools**. When cleared, the Data File Tools (Technology Preview) version is used, when possible. When this editor does not support the action you are attempting to complete, Classic Data File Tools is used instead. This check box is selected by default.

Database Access

Visual COBOL version 2.3 provides the following enhancements to database access:

COBSQL

Visual COBOL version 2.3 provides:

- Selection and configuration of the Oracle Pro*COBOL preprocessor for compiling COBSQL applications in project properties on the **SQL** tab.
- Support for COBOL directives SOURCEFORMAT=TERMINAL and SOURCEFORMAT=VARIABLE for Pro*COBOL applications.

HCO for DB2 LUW

Visual COBOL version 2.3 provides:

- Support for MFHCO mode across all platforms by default via the new HCO "NOHCO" DB2 compiler directive option. See the *HCO* DB2 compiler directive option topic for details.
- A new DB2 compiler directive option, OPTPER "NOOPTPER", that enhances performance for CHARSET EBCDIC processing. See the *OPTPER* DB2 compiler directive option topic for details.
- A new DB2 directive option, BINDDIR, which specifies an alternative directory in which to write the DBRM file created during compilation. See the *BINDDIR* DB2 compiler directive option topic for details.

OpenESQL

ADO.NET Connection Editor

In this release:

- The ADO.NET Connection Editor has been redesigned using a series of Wizards that guide you through the processes of adding, copying, and removing connections.
- Context Help is now provided for the main window, and each Wizard page and dialog box.

Date/Time Processing

This release provides streamlined datetime processing for ODBC and ADO.NET.

Performance

This release includes a new SQL compiler directive option, OPTPER "NOOPTPER", that enhances performance for CHARSET EBCDIC processing. See the *OPTPER* SQL compiler directive option topic for details.

PL/I

This version provides 64-bit support for PL/I on appropriate platforms. See *Additional Software Requirements* for details.

PostgreSQL

In this release, PostgreSQL 9.4 has been tested with OpenESQL and OpenESQL Assistant with the following PostgreSQL software:

Server software PostgreSQL EnterpriseDB version 9.4.1-3

- Client software**
- psqLODBC driver version 09.03.04.00
 - Npgsql ADO.NET 4.0 driver version 2.2.5

PostgreSQL 9.4 has been tested with OpenESQL and OpenESQL Assistant on the following Windows platforms:

- Windows 32-bit
- Windows 64-bit



Note: Micro Focus provides compatibility for PostgreSQL but does not directly contribute to or support the PostgreSQL open source project. Any issues relating to PostgreSQL functionality should be addressed through an open source support vendor.

SQL Server Visual COBOL version 2.3 provides support for the SQL Server OUTPUT clause.

XA Switch Modules

In this release, the XA interface has been redesigned to provide:

- Consistent look and feel for SQL Server, DB2, and Oracle user personalization
- Consistent look and feel for both RM dynamic and static registration (SQL Server, DB2, Oracle, generic one-phase commit)
- Additional support for two instances of the same switch module using Web Services applications via the new XAID compiler directive
- Using a specified XA resource only with batch applications executing under Enterprise Server

File handling

This release contains the following new configuration options:

ACUFH Enables or disables the use of the ACU file handler (ACUFH), which is required to handle Vision and RM/COBOL indexed files.

ESACUFH Enables or disables the use of the ACU file handler (ACUFH) for file handling operations running under Enterprise Server. ACUFH must also be enabled for this option to take effect.

Library routines

The following library routines are new in this release:

CBL_MANAGED_SESSION_GET_USERDATA Retrieves user data saved in the current RunUnit.

CBL_MANAGED_SESSION_SET_USERDATA Sets user data in the current RunUnit.

The following library routines contain new parameters in this release:

CBL_LOCATE_FILE You can now specify a file name that is a null-terminating string, which has resulted in three new values available for the `user-mode` parameter.

Managed COBOL syntax

The following enhancements have been made to the managed COBOL syntax:

- The `TYPE OF type-name[ANY...]` syntax enables you to obtain the `System.Type` (.NET) or `java.lang.Class` (JVM) object for a generic class, interface, or delegate.
- The `self::` or `super::` syntax is no longer required to access inherited data within a subclass.
- The `ATTRIBUTE-ID` syntax enables you to define new attribute types, which can be used in various contexts.

Micro Focus Infocenter

The Micro Focus Infocenter Web site (<http://documentation.microfocus.com>) has been upgraded and now includes the following improvements:

- Scope being persisted when you select a product documentation in the Product Documentation section on the Micro Focus SupportLine Web site and choose to view the documentation in the Micro Focus Infocenter.
- Updated **Scope** settings - provides the ability to nest four levels deep when setting a scope.
- Scope being persisted between browser sessions once it has been set.
- Creating automatic scopes using the **Search Topics** icon, .

- A link to change the scope from the search results when there are too many results.
- Improved Boolean search expressions.
- Details included with the search results.
- Help on how to use the Infocenter and how to construct search expressions - available using the Infocenter Help button, .

Micro Focus Unit Testing Framework



Note: This is a technology preview feature only. It is being made available to allow you to test and provide feedback on this new capability, but it is not intended for production use and is not supported as such. Furthermore, Micro Focus does not guarantee that this feature will be delivered at a GA level and if it is, then the functionality provided might differ considerably from this technology preview. During the preview, you are encouraged to share your feedback and experiences via the Micro Focus community forum - <http://community.microfocus.com/microfocus/>.

The Micro Focus Unit Testing Framework is an xUnit style testing framework, available from the command line, for procedural COBOL applications.

It includes much of the architecture you would expect in an xUnit framework. The test runner is a 32- or 64-bit executable that you run from a Visual COBOL command prompt. A test fixture or suite is a COBOL program compiled to `.dll` that can include the setup, the test case code, and the teardown associated with the test case.

Test results are available in a number of formats. By default, results are displayed to screen and to a `.txt` file, but you can use additional parameters on the command line to produce reports in JUnit format.

Microsoft Azure support

Visual COBOL support for Microsoft Azure has been updated to version 2.6 of the Microsoft Azure SDK.

Support has been added to the product for making any future versions of the Microsoft Azure SDK available before the next major release of Visual COBOL. Micro Focus will deliver support for these only upon customers' requests.



Note:

2.3 HotFix 1 update: This HotFix provides support for version 2.7 of the Microsoft Azure SDK in Visual COBOL for Visual Studio 2013 and 2015.

Personal edition licensing

This release includes a Personal Edition licensing option for Visual COBOL for Visual Studio 2015.

Preprocessors

Support has been added in the IDE for enabling and using multiple preprocessors with your projects.

A new page, **Preprocessors**, has been added to the project's and the files' properties of native COBOL applications to enable you to choose one or more preprocessors to use when building your application and to specify their order of execution.

New reporting capability is now available for user preprocessors: `resp-main code 18` indicates that a buffer contains a data name to be marked as modified by the immediately preceding preprocessed line. The data name may be qualified and `resp-more` contains the column information for the reference.

Profiler

Visual COBOL now provides support for Profiler for native COBOL applications directly from within the IDE. To produce reports, you need to:

1. Enable Profiler in the COBOL property page for a project.

2. Compile your application to apply the changes.
3. Run your application with Profiler to produce the relevant reports.

REST service interfaces

RESTful service interfaces utilizing JSON as the media type in request and response messages are now supported using the Interface Mapping Toolkit. This enables you to extend COBOL applications using modern transport payloads and protocols.

RM/COBOL Compatibility

This release includes improved support for RM dialect applications. Please consult with Micro Focus before considering a transition from RM/COBOL to Visual COBOL.

Single file support

The recommended way to work with files within Visual COBOL is to include them in a project. For situations where you might want to quickly open edit a single file, Visual COBOL now provides support for native COBOL files in the IDE when the file is not opened as part of a project. There is limited support for the IDE editing, compiling and debugging features as full support requires a project file.

To enable full IDE support for single files, Visual COBOL provides a path for creating projects from them - right-click such files in the editor, and click **Create COBOL Project**.

Tunables

Visual COBOL version 2.3 contains the following updates to tunables:

- default_cancel_mode** A new parameter, and default, has been introduced for this tunable; see *default_cancel_mode* for more information.
- subsystem_cancel_mode** A new parameter has been introduced for this tunable; see *subsystem_cancel_mode* for more information.

Updated run-time system

COBOL Server has been updated to provide an execution environment capable of running applications that were each built using different development products. A consequence of this is that If your application has a main COBOL executable (.exe) that was built with a previous version of Visual COBOL, you should ensure that the executable is rebuilt and packaged with the new run-time system. You can rebuild from the IDE or the command line.

Other COBOL subprograms built with previous versions of Visual COBOL are not required to be rebuilt.

What was New in Visual COBOL 2.2 Update 2

Visual COBOL 2.2 Update 2 provided enhancements in the following areas:

- [Visual Studio](#)
- [Character Set Enhancements](#)
- [Code Analysis](#)
- [Database Access](#)
- [Micro Focus COBOL enhancements](#)
- [External Security Facility \(ESF\)](#)
- [Enterprise Server MQ-IMS Bridge](#)
- [Tunables](#)

Visual Studio

Visual COBOL 2.2 Update 2 provides enhancements in the following areas:

- Call Hierarchy - support for the Call Hierarchy window has been enhanced and it now shows types and members across the entire solution.
- Debugging enhancements - includes enhanced support for the Autos windows for native and managed COBOL, support for querying EBCDIC data in managed code, and support for the standard visualizers for data items and groups in native COBOL.
- Expanded Copybook View (only supported in Visual Studio 2012 and Visual Studio 2013) - now supports Intellisense, collapsing and expanding of outlining regions, and code snippets.
- Find All References - you can now configure the scope of the Find All References command. The default behavior is to search for references in the current COBOL project and now, you can enable Find All References to search for matches in all managed COBOL projects that are part of your solution.
- Net Express Project Import wizard - now includes a number of usability enhancements.
- Run-time configuration - you can now use the application configuration file, `Application.config`, in native COBOL projects to set all run-time tunables.

Character Set Enhancements

The following character sets, available using the MFCODESET environment variable, have been enhanced or added in this release:

- Thai Extended (0066) - new
- Korean (0082)
- Simplified Chinese (0086)
- Traditional Chinese (0886)

There are also a number of double-byte character sets that are now capable of mixed single-byte and double-byte character conversion; see the definition of MFCODESET in *Environment Variables in Alphabetical Order* for more information.

Code Analysis

Visual COBOL for Visual Studio can produce the following COBOL reports:

- Dead Code - finds unreferenced items or any piece of code that can't be reached during execution.
- Unreferenced Data - finds any data items that are not explicitly referenced in the Procedure Division of a program.
- Undeclared Procedures - finds any procedures that are referred to but not defined.
- Copybook Structure - displays the hierarchy of any copybooks defined in a program.
- Program Statistics - provides general information, such as number of source code lines, number of data items, and size of data items.
- Unexecuted Procedures - finds any procedures that are defined but not referred to.

Database Access

The following new features are available in database access support:

COBSQL

In Visual Studio:

- Pro*COBOL support - you can now select and configure the Pro*COBOL COBSQL preprocessor for compiling COBSQL applications on the **SQL** tab in the project's properties.
- KEEPCOMP - the new KEEPCOMP directive resolves COMP/COMP-5 issues with Oracle applications on little-endian platforms.

HCO for DB2 LUW Visual COBOL version 2.2 Update 2 introduces GEN-HV-FROM-GROUP - a new DB2 ECM compiler directive option, that generates host variables for all elementary data items when a multiple-level group variable is used in a FETCH or singleton SELECT DB2 statement.

OpenESQL This version provides the following new OpenESQL features:

- Support for SQL Server 2014.
- New SQL Compiler directive options:
 - DETECTDATE=SERVER - resolves host variables alignment with column data types in an SQL table.
 - GEN-HV-FROM-GROUP - generates host variables for all elementary data items when a multiple-level group variable is used in a FETCH or singleton SELECT SQL statement.
- Sample applications - the following native COBOL SQL sample applications are new with this version:
 - Get Diagnostics - demonstrates how to use GET DIAGNOSTICS EXEC SQL calls to get diagnostic information from various DBMSs.
 - LOB Data Types - Demonstrates how to INSERT and SELECT LOB data in a native application using various DBMSs.

XA switch modules



Restriction: This feature applies only when the Enterprise Server feature is enabled.

The following XA switch module updates are available in this version:

- Oracle switch module:
 - Supports User Impersonation when statically registered.
 - Enables you to specify which XA resource definitions use User Impersonation.
 - Now compiled with one source file, rather than two.
- SQL Server switch module:
 - Enables you to specify which XA resource definitions use User Impersonation.
 - Now compiled with one source file, rather than two.

Micro Focus COBOL enhancements

The following enhancements have been made to Micro Focus COBOL:

- The following phrases have been added to the XML GENERATE statement:
 - NAME
 - TYPE
 - SUPPRESS
- The following intrinsic functions have been added:
 - ULENGTH
 - UPOS
 - USUBSTR
 - USUPPLEMENTARY
 - UVALID
 - UWIDTH

External Security Facility (ESF)

The Enterprise Server External Security Facility (ESF) now supports caching the results of some security queries. This can improve the performance of enterprise server instances and of the MFDS when they are configured to use external security.

To enable caching, you need to set non-zero values for the **Cache limit** (maximum size of the cache) and **Cache TTL** (Time To Live, or how long before a cached result expires) settings on the **MFDS Security** tab, the **Default ES Security** tab, or on the **Security** tab for an individual enterprise server. (Currently, the cache settings for Security Managers have no effect; you need to set cache parameters on one of the three Security pages mentioned earlier.)

For more information, see <http://supportline.microfocus.com/examplesandutilities/doxygen/caching.html>.

Enterprise Server MQ-IMS Bridge

At Visual COBOL 2.2 Update 2 the Enterprise Server MQ-IMS Bridge was supported at GA level. It had previously (from Visual COBOL 2.2 Update 1) been available as a Technology Preview item only.

Tunables

Visual COBOL 2.2 Update 2 includes the following new tunable:

- `reduce_java_signals` - specifies the options that are passed to a JVM when mixing Java and COBOL.

What was New in Visual COBOL 2.2 Update 1

Visual COBOL 2.2 Update 1 provided enhancements in the following areas:

- [Visual Studio IDE](#)
- [Micro Focus Heartbleed Update](#)
- [ACUCOBOL-GT Compatibility](#)
- [COBOL Source Information](#)
- [Compare and Synchronization Monitor](#)
- [Compiler Directives](#)
- [Database Access](#)
- [Enterprise Server Integration in the IDE](#)
- [Enterprise Server MQ-IMS Bridge \(Technology Preview\)](#)
- [Environment Variables](#)
- [Fileshare Recovery](#)
- [Line Numbering for COBOL Programs](#)
- [Managed COBOL Syntax](#)
- [Rumba Integration with Visual Studio](#)
- [Run-time Launch Configuration Files](#)

Visual Studio IDE

Visual COBOL 2.2 Update 1 provided the following enhancements to COBOL support in the IDE:

- Expanded Copybook View** (Visual Studio 2012 and 2013 only.) The Expanded Copybook View functionality in the editor was added in the 2.2 release and this was enhanced in 2.2 Update 1 so it is now possible to debug into the expanded copybooks. The functionality includes:
- Stepping inline in the expanded copybook view when debugging - when stepping into or hitting a breakpoint in the copybook, the copybook automatically expands. You can disable this feature from the IDE preferences.
 - Showing Pinned Data tips inside the expanded copybook.
 - Support for **QuickWatch** and **Add Watch** inside the expanded copybook.

- Executing some of the debugger commands from the context menu from the expanded copybook view - for example, **Show Next Statement**, **Run To Cursor** and **Set Next Statement**.

Some limitations apply.

Call Hierarchy	(Visual Studio 2012 and 2013 only.) Visual COBOL now supports the Visual Studio Call Hierarchy window for analyzing COBOL PERFORM statements.
Find All References	The Find All References functionality now works for managed OO COBOL code.
Navigation	<p>You can now use Alt+Shift+Up Arrow (or Down Arrow) to navigate the variables in the editor.</p> <p>You can use the drop-down types and members to navigate to the different sections (Linkage section, File section etc.) in your source files.</p>
Project Details Window	The Project Details Window now includes columns showing the COBOL dialect of the files and the SQL properties set on them.
Run-Time Configuration	<ul style="list-style-type: none"> • You can now use the application configuration file (<code>application.config</code>) in native COBOL projects to specify some additional settings such as the search order for called programs, command line handling, or file handling. You can do this from the new Run-Time Configuration tab. • When an executable is built using the IDE, the application configuration file is automatically copied to the output folder and is renamed as <code><executable-base-name>.exe.mfgcf</code>.
Samples	<p>Visual COBOL 2.2 Update 1 includes the following new samples:</p> <ul style="list-style-type: none"> • Airport Demo (managed) - shows how to create a COBOL WCF REST service and then use a client application to consume it. • Airport Demo (native) - a basic lookup program that reads the information about airports from a .dat file and outputs the distance between two airports. <p>The Sandcastle sample has been updated and now uses Sandcastle Help File Builder v1.9.8.0.</p>
Solution Explorer	<p>(Visual Studio 2012 and 2013 only.) Support is now available in Solution Explorer for the following features:</p> <ul style="list-style-type: none"> • Errors and warnings filters - click the arrow next to the  (Pending Changes Filter) icon in the Solution Explorer toolbar, and either click  Errors Filter or  Errors and Warnings Filter to show the files that result in errors or also cause warnings. • Searching in Solution Explorer - You can use the search field in Solution Explorer to search for files in your solution. Searching also finds copybooks in the copybook dependency view even if they are not part of any of the projects in the solution.

Micro Focus Heartbleed Update

The OpenSSL library used in this product was updated to version 1.0.1g to fix the "Heartbleed" vulnerability with TLS heartbeat requests.

ACUCOBOL-GT Compatibility

The following ACUCOBOL-GT support has been added in this release:

-Di compiler option The -Di compiler option, which initializes Working-Storage data items based in their type, is now supported.

COBOL Source Information

The **Quick Browse** option is now available as a context menu command in the editor.

Compare and Synchronization Monitor

With the release of Visual COBOL 2.2 Update 1, the Compare and Synchronization Monitor has been updated to version 2.

Version 2 is greatly improved in terms of performance, especially during initial checkout of partitioned data sets or when synchronizing a large number of members. Also, the user interface has been improved, and some of the functions available in the old version have now changed or become obsolete.

Compiler Directives

The following Compiler directives have been added in this release:

ILPARAMS Determines the way in which you call a method that contains an array as its last receiving parameter.

INIT-BY-TYPE Initializes Working-Storage Section data items to a default value, according to their type.

- Alphabetic, alphanumeric, alphanumeric edited, and numeric edited items are initialized to spaces.
- Numeric items are initialized to zero.
- Pointer items are initialized to null.
- Index items are initialized to the value 1.

Database Access

The following new features have been added as part of database access support:

DB2 ECM

- Support added for DB2 LUW version 10.5.
- Enhanced RETURN-CODE processing.

OpenESQL

- Enhanced internationalization support for UNICODE, DBCS and MBCS.
- Enhanced GET DIAGNOSTICS statement support.
- Enhanced LOB support for CLOB, BLOB and DBCLOB data types.
- ADO.NET connection editor now provides context help.
- Enhanced IDE support for OPTION directives.
- Now provides support for the creation of save points and rolling back to save points.

XA Switch Modules



Restriction: This feature applies only when the Enterprise Server feature is enabled.

- New two-phase commit module for SQL Server based on Microsoft's XA switch. This provides support for xa_recover.
- Support for DB2 LUW version 10.5.
- Support for Oracle version 12.1.

Enterprise Server Integration in the IDE

You can now use the context menu for the servers in Server Explorer to enable the display of the Enterprise Server log information in the Output window.

Enterprise Server MQ-IMS Bridge (Technology Preview)



Note: At Visual COBOL 2.2 Update 1 this was provided as a technology preview feature only. It was made available to allow you to test and provide feedback on this new capability; however, this feature was not intended for production use and was not supported as such.

Visual COBOL version 2.2 Update 1 provided support that enables WebSphere MQ applications to communicate with IMS applications in an Enterprise Server region.

Environment Variables

The following environment variable has been added in this release:

strictvsam strictvsam enables strict mainframe emulation when processing VSAM files.

When set to ON and running under mainframe emulation, file status 37 is returned for an existing VSAM file when opened for OUTPUT if the file has data or previously had data written to it, or if the file is of a different format to the file on disk. When set to OFF, file status 0 is returned and a new file is created when an existing VSAM file is opened for OUTPUT. This variable is set to OFF by default.

Fileshare Recovery

Recovery of Fileshare data files has been enhanced.

Rollback recovery is a faster process that aims to fix the files from their failed state.

This process cannot be used in all scenarios, but a new user exit has also been introduced that allows you to programmatically control which files you wish to recover with this process.

Hot backups are also a new introduction, which allow you to perform a backup without having to shut down Fileshare.

Line Numbering for COBOL Programs

Visual COBOL version 2.2 Update 1 provided options for auto-inserting or removing line numbers in source files open the editor. Features include:

- COBOL numbering - line numbers are inserted in the sequence area of the code (columns 1 - 6), starting by default at 000100 at the first line, incrementing by 100 by default.

Micro Focus recommends that you use COBOL numbering only if your files are in fixed or variable source format.

- Standard numbering - line numbers are inserted immediately to the right of area B, in columns 73 - 80, starting by default at 00000100 at the first line, incrementing by 100 by default.

Micro Focus recommends that you use Standard numbering only if your files are in fixed format.

- The **Renumber** and **Unnumber** commands available from the context menu in the editor.

Managed COBOL Syntax

Visual COBOL version 2.2 Update 1 includes the following enhancements to the managed COBOL syntax:

Specifying parameters in the method signature You can now specify passing parameters and returning items in the method signature, instead of using a Procedure Division header. This applies to methods, indexers, iterators, constructors and delegates.

CONSTANT keyword Use the CONSTANT keyword on a field to protect it from being altered.

Operations on string fields You can now use the STRING, UNSTRING and INSPECT statements on fields of type string.

Rumba Integration with Visual Studio

Visual Studio now offers context menu commands for launching the Rumba mainframe display (Desktop or embedded) from Solution Explorer or from Server Explorer. The embedded Rumba display now provides keyboard mapping and color settings.

You can use the display of choice both in debugging and when running applications.

Run-time Launch Configuration Files

Use a run-time launch configuration file to ensure an application can be launched when it is deployed in a separate location to the run-time system (in the case of dynamically bound applications), or when the licensing daemon is not already running.

What was New in Visual COBOL 2.2

Visual COBOL 2.2 provided enhancements in the following areas:

- [Enhancements for Visual COBOL 2.2 for Visual Studio 2012](#)
- [ACUCOBOL-GT Compatibility](#)
- [RM/COBOL Compatibility](#)
- [Application Configuration](#)
- [COBOL Source Information \(CSI\)](#)
- [Compiler Directives](#)
- [Consolidated Tracing Facility](#)
- [Converting Net Express Projects](#)
- [Enhanced Accept and Display Statements](#)
- [Debugging](#)
- [Grouping Files in Virtual Folders in Solution Explorer](#)
- [File Handling](#)
- [Interface Mapping Toolkit](#)
- [Managed COBOL](#)
-
- [Searching in Copybooks](#)
- [COBOL Editor Enhancements](#)
- [Upgrading from Net Express to Visual COBOL](#)
- [WCF REST Service Application projects](#)
- [Microsoft Windows Azure](#)
- [XML Extensions](#)

Enhancements for Visual COBOL 2.2 for Visual Studio 2012

The following enhancements are applicable to Visual COBOL for Visual Studio 2012:

- Copybook Dependencies - Solution Explorer shows a tree view of the dependencies of the COBOL programs and copybooks on copybooks in a project to help you navigate easily around the source code.
- Expanded Copybook View: Technical Preview - The COBOL editor supports showing the contents of copybooks inline at the place where they are referenced in a program or a copybook. Features include:
 - Support for copybooks referenced using any of the following statements - COPY, -INC, ++INCLUDE, EXEC SQL INCLUDE.
 - Support for editing copybooks in the expanded copybook view (except for copybooks shown with replaced values or such for which the user does not have write access).
 - For copybooks referenced using a COPY... REPLACING statement, you can show either the original contents of the copybook file or show its contents with replaced values. The expanded copybook view with replaced values is read-only which is indicated by a yellow background color.



Note: You can configure the background color of the read-only view from **Tools > Options > Environment > Fonts and Colors**, and changing the background color for the **COBOL Read-Only Background** item.

- **Go To Definition, Find All References**, navigation using the drop-down types and members, and navigating to an error from the **Error List** window all find results or locate information in the expanded copybook view.
- Support for Intellisense and code snippets when editing copybooks in the expanded copybook view.
- Support for the Visual Studio outlining feature (collapsing and expanding regions of code) within the expanded copybook view and for regions that include expanded copybooks.



Note: The following limitations apply to the expanded copybook view:

- Bookmarks are not supported in the expanded copybook view. You can add bookmarks in copybooks that are opened as standalone files.
- It is not possible to edit copybooks referenced through a COPY... REPLACING statement in expanded copybook view when the view shows the replaced values. In this case, the expanded copybook view is read-only.
- The read-only status of expanded copybooks is not preserved when you close the project. Next time you open the project, the copybook is shown in the Expanded Copybook view but not as read-only.
- If you used the OF or IN phrases with a COPY statement, it is not possible to show the corresponding copybook in the Expanded Copybook view or open it using the **Open "CopybookName" in New Window** command if the copybook is stored in a library file (where you have used the COPYLBR Compiler directive).
- Navigating to items in the expanded copybook view is not supported from the **Class View** or using the **Navigate To** command.
- References to file locations in the output windows such as pointing to the build output, or in the results received by **Find in Files**, do not point to items in the expanded copybook view.
- Selecting the currently selected statement in the call stack in an expanded copybook may show the actual copybook. You can use the **Show Next Statement** command of the debugger to return the actual expanded copybook position.
- The Breakpoints window always shows the line number of a breakpoint within the file containing it, not the line number within expanded copybook view.
- The **Call Stack, Threads** and **IntelliTrace** windows all display line numbers corresponding to the file containing the statement, not the line number within expanded copybook view.
- If you move the caret to an invalid position in the expanded copybook view, the **Set Next Statement** command will display an 'Unable to set the next statement to this location' error message and the copybook will be opened as a separate document.
- When stepping through managed code, if the next statement to step onto is within the same class or program, and on the same line number, but in a different source file, the debugger steps over that statement. For example, if you try to step from line 10 of `Program1.cb1` to line 10 of `Copybook1.cpy`, the debugger steps over the statement in the copybook.

ACUCOBOL-GT Compatibility

The following enhancements are applicable to Visual COBOL:

- Accessing data files through AcuServer - You can now access your ACUCOBOL-GT data files, both sequential and Vision files, through AcuServer.
- Standard library routines - Support for the following library routines has been added:
 - C\$GETPID
 - C\$JUSTIFY
 - C\$LIST-DIRECTORY
 - C\$LOCKPID

- C\$REGEXP
 - C\$RUN
 - C\$SLEEP
 - C\$SYSTEM
 - C\$TOLOWER
 - C\$TOUPPER
 - I\$IO
- Using Vision files with Micro Focus Data File Tools - You can now use some of the Data File Tools functionality with Vision files. You can:
 - Convert Vision files to Micro Focus format using the Data File Converter and the DFCONV command line utility.
 - Edit Vision files using the Data File Editor.



Note: For more information about the **Data File Tools** utility, see *Data Tools*.

RM/COBOL Compatibility

The following support has been added to Visual COBOL in this release:

- Subprograms - Support for the following subprograms (referred to as library routines in Visual COBOL) has been added:
 - C\$OSLockInfo
 - C\$SecureHash
- recover1 - The recover1 utility, RM/COBOL's indexed file recovery utility, is now distributed with Visual COBOL. Refer to the *RM/COBOL File Handling* section of *RM/COBOL Compatibility* for details of its use.

Application Configuration

You can now set environment variables for when you run native projects from within the IDE from the project's properties - click **Environment** on the **Application** tab in the project properties.

COBOL Source Information (CSI)

COBOL Source Information (CSI) provides a quick and easy way of providing you with information about your program when you are working on it. You enter a query in the **Quick Browse** dialog box, the CSI query control and CSI returns the results of the query in the Micro Focus Code Analysis window.

Compiler Directives

The following Compiler directives are new:

ACU-UNDERSCORE	This directive treats underscores in COBOL words as hyphens.
ILSHOWPERFORMOVERLAP	This managed COBOL-only directive generates a warning when an overlapping PERFORM range is detected in the program.
ILEXPONENTIATION	This managed COBOL-only directive enables you to optimize exponential arithmetic operations by specifying the calculation method used.
EXITPROGRAM	This directive determines how the EXIT PROGRAM statement is executed.

The following Compiler directives have changed

CHANGE-MESSAGE The scope of this directive has been widened to allow you to change the severity of different types of error messages, not just syntax checking messages.

- DIALECT"RM"** DIALECT"RM" now sets PERFORM-TYPE"RM". If you recompile an application that uses DIALECT"RM", the behavior may change for nested PERFORM statements. If that is the case, explicitly set PERFORM-TYPE"MF" after DIALECT"RM" to continue with the previous behavior.
- HIDE-MESSAGE** The scope of this directive has been widened to allow you to hide different types of error messages, not just syntax checking messages.
- PRESERVECASE** This directive now defaults to PRESERVECASE when compiling native COBOL; managed COBOL compilation already defaults to PRESERVECASE. This results in externally visible identifiers preserving their case instead of being converted to uppercase.

Consolidated Tracing Facility

The following changes have been made to the Consolidated Tracing Facility (CTF):

New properties and variables for existing emitters The following support has been added to existing emitters.

Properties The following property has been added to the BINFILE emitter:

Property	Description
RunUnitID	Controls whether the RunUnit information is included in the trace.

Variables Four new pseudo-variables for the FILE property have been added to the BINFILE and TEXTFILE emitters:

pseudo-variable	Description
\$(PLATFORM)	A platform specific constant, useful when two run-time systems are in the same process, and you require separate trace files
\$(RUNUNIT)	A unique number that represents the managed RunUnit ID
\$(RUNUNIT_SESSIONNAME)	The session name passed to the managed RunUnit
\$(RUNUNIT_GUID)	The globally unique identifier associated with the managed RunUnit

Converting Net Express Projects

Conversion of Net Express projects has been enhanced. You now use the Visual Studio conversion wizard to import and convert Net Express projects (.app files) to Visual Studio solutions.

Enhanced Accept and Display Statements

Two of the existing Enhanced ACCEPT and DISPLAY settings available through Adis have additional values, which are aimed at RM/COBOL users migrating their source code to Visual COBOL. The new values are:

- Emulation of RM/COBOL-85 style data entry for numeric data entry on ACCEPT statements.
- Emulation of an RM/COBOL backspace in free format fields when in replacement editing mode, in that deleted characters are removed and characters to the right are shifted left, the same as when in insertion editing mode.

For more information on how to set these values, refer to *Configuring Enhanced ACCEPT and DISPLAY*.

Debugging

Displaying debug information for managed applications You can set the DEBUG constant for managed COBOL projects on the **COBOL** tab in the project properties. This enables you to use the System.Diagnostics.Debug class in your applications to ensure they write diagnostic information in the Output window for projects compiled for Debug but not for projects compiled for Release.

Changing the display format for individual items in the Watch window It is now possible to change the display format for individual items in the Watch window in COBOL. To do this, click a row, press **F2**, and type: *Variable,h* or *Variable,x* to always display the values in hexadecimal format; *Variable,d* to always display the values of variables in decimal format, and of strings - as text.

Grouping Files in Virtual Folders in Solution Explorer

Visual COBOL now provides a Virtual View of a project within Solution Explorer. In the Virtual View you use virtual folders to improve navigation by logically grouping the files that make up the project. You can also create your own virtual folders to group files of your choice (a file can only belong to one virtual folder). The files can be of different file types.

File Handling

New features include:

- Converting and editing Vision and RM/COBOL indexed data files using the Data File tools is now supported.
- Access to data files (either sequential or indexed) through AcuServer is now supported.
- Access to Vision and RM/COBOL indexed data files through Enterprise Server is now supported.

Interface Mapping Toolkit

Visual COBOL now supports the creation and deployment of COBOL program-based services using the Interface Mapping Toolkit (IMTK).

Managed COBOL

Documentation A guide that provides a basic introduction to Object-Oriented Programming (OOP) for COBOL developers, *An Introduction to Object-Oriented Programming for COBOL Developers*, with examples is now available from the *Product Documentation* section on the Micro Focus SupportLine Web site - [click here to download it](#).

Procedural Multi-Output Project type A new .NET managed project template, Procedural Multi-Output Project, is now available. The project compiles procedural programs to individual managed assemblies or executables as opposed to the standard managed project templates that produce a single assembly. This project type is useful when you are moving existing procedural applications that consist of multiple programs to Visual COBOL and to managed code. When calling a subprogram, the COBOL run-time system can locate the assembly using the name of the called program which will match the assembly file name. In this case, the assembly does not have to be preloaded. See *Types of COBOL Project Templates* for more details and for a list of the limitations of this project type.

Named and optional parameters Two new types of parameter have been introduced for use during method invocation:

Named parameters	As part of the invocation expression, you can define a value for a parameter named in the method definition. The named argument
-------------------------	---

must be specified after any positional arguments, and must not correspond to any of those preceding arguments.

Optional parameters

Optional parameters are parameters defined with a default value in the procedure division header of the invoked method. If none of the arguments passed in during invocation correspond to this parameter, the default value is used in the method; if an argument does correspond, the value that was passed in is used.

Delegates and events

A number of new features have been added that relate to delegates and events:



Note: Some of these features were also available in previous versions of Visual COBOL.

The ATTACH and DETACH statements

Use these statements to attach or detach a delegate, method group or an anonymous method to or from an event.

The RUN statement

Use this statement to invoke a delegate once it has been created.

Combining delegates

Use the '+' operator to add a method group, anonymous method or another delegate to a delegate, and use the '-' operator to remove a method or another delegate from a delegate.

Method groups conversions

Use the METHOD keyword to specify a compatible method from a method group, and convert it to a delegate.

Searching in Copybooks

The search in copybook files has been enhanced. Visual Studio's **Find in Files** now enables you to perform a search in all copybooks - the ones that are part of the project, and in the ones that are found in the paths defined on the **Dependency Paths** tab in the project's properties. To do so, set the search scope in the **Find and Replace** dialog to **COBOL Project Copybook Paths**.

COBOL Editor Enhancements

Navigating in the editor is now more similar to the editor navigation in Net Express. You can configure things like word wrap and indentation, the movement of the cursor using the **Home** and **End** keys, the style of the ruler and the colors of the text and the margins:

Smart editing mode

The default mode in the editor is now Smart edit mode. It controls the word wrapping and the indentation in the different COBOL areas. You can configure the Smart edit mode from the **Tools > Options > Text Editor > Micro Focus COBOL > Margins** page. To toggle the mode, click  or  (in Visual Studio 20122013) in the COBOL toolbar.

Home and End keys navigation

You can configure how the **Home** and **End** keys move the cursor in the editor on the **Tools > Options > Text Editor > Micro Focus COBOL > Margins** page. To toggle the mode, click  or  (in Visual Studio 20122013) in the COBOL toolbar.

Configuring the ruler

You can specify whether to display the horizontal ruler in the editor and also choose its style, a default style or a mainframe style for which the ruler indicates COBOL area A/B, from **Tools > Options > Text Editor > Micro Focus COBOL > Margins** and check **Show ruler**.

Support for SOA



Restriction: This topic applies only when the Enterprise Server feature is enabled.

Visual COBOL now includes support for creating Web service and Enterprise Java Bean applications using the Interface Mapping Toolkit (IMTK) in conjunction with Enterprise Server. If you are upgrading to this release from an earlier version of Visual COBOL, you may need to apply for a new authorization code in order to access the functionality - please contact Micro Focus SupportLine to receive an updated authorization code. Note that the Visual COBOL Personal Edition license does not support the IMTK functionality.

Upgrading from Net Express to Visual COBOL

A new section in the product help, *Upgrading from Net Express to Visual COBOL for Visual Studio 2017*, provides guidance on how to move existing applications either developed or debugged in the Net Express IDE into the Visual Studio IDE.

WCF REST Service Application projects

Visual COBOL now provides a project template for creating COBOL WCF REST Application services. The requirements are - Visual Studio Professional, Premium, or Ultimate (Visual Studio Shell does not support WCF applications), and versions of the .NET Framework 4.0 and later. For more information about WCF and REST, check Microsoft's MSDN.

Microsoft Windows Azure

Visual COBOL for Visual Studio now enables you to create, build, debug, run, and package COBOL applications for use with Windows Azure. It provides:

- Project template for a COBOL cloud service. This creates a solution containing a project for the cloud service and projects for the web and worker roles you specify.
- Demonstration projects, showing a web and worker role in COBOL, with the business logic remaining in unchanged COBOL programs.
- Help. This includes a tutorial that shows how to create a simple cloud service for COBOL and how to deploy and run the service as an off-premises Microsoft Azure instance.

This functionality was previously available as an AddPack but is now integrated into Visual COBOL.

XML Extensions

You can now use XML Extensions in your managed COBOL projects.

Use XML Extensions to import and export XML documents to and from COBOL working storage. Specifically, XML Extensions allows data to be imported from an XML document by converting data elements (as necessary) and storing the results into a matching COBOL data structure. Similarly, data is exported from a COBOL data structure by converting the COBOL data elements (as necessary) and storing the results in an XML document.

While importing or exporting data to or from XML documents, you can apply XSLT transforms to the data by using XSLT stylesheets.

For more information, refer to the XML Extensions User's Guide, available from the product documentation section of the SupportLine website (<http://SupportLine.MicroFocus.com/ProductDoc.aspx>)

What was New in Visual COBOL 2.1 Update 1

Visual COBOL 2.1 Update 1 provided enhancements in the following areas:

- [Compiler Directives](#)

- [DB2 ECM](#)
- [Copybooks Enhancements](#)
- [Enterprise Server](#)
- [DB2 ECM](#)
- [Debugging Enhancements](#)

Compiler Directives

You can now set SQL Compiler directives and their values more easily, using a table of tick boxes in a project's Properties dialog box.

DB2 ECM

- Support for 64-bit DB2 ECM
- Support for 64-bit compile and runtime
- Support for DB2 10.1
- New DB2 SQL compiler directive option, BGP, to enable background parsing

Copybooks Enhancements

For variables in a copybook that are modified by COPY... REPLACING statements in your code, the Autos window displays all values defined in the source code. When there are multiple COBOL programs in your project that perform a COPY... REPLACING in a copybook, the Autos window only uses the replacing values found in the first COBOL program. In addition, a new command, **Open copybook with replaced values**, is now available from the editor for the copybooks that the COPY... REPLACING statements modify.

Enterprise Server

The following new features and enhancements are available:

Clustering	COBOL Server Clustering allows the scaling-out of work units, so that an increased number of operating system images can share the workload, resulting in high-performance, multi-system data sharing across all platforms.
Historical Statistics Facility	The Historical Statistics Facility has been extended to include the generation of JCL file records, increasing the amount of information customers have available to assist them in monitoring and tuning their COBOL Server installations.
Recovery of in-doubt XA transactions	Some events in XA environments can result in 'in-doubt' transactions, where all parts of a composite transaction are not committed through all participating resource managers. The recovery of such in-doubt transactions is now supported.
SSL Support for the CICS Web Interface	COBOL Server now allows clients and servers to identify themselves through X.509 certificates and participate in SSL-enabled conversations.

DB2 ECM

- Support for 64-bit DB2 ECM
- Support for 64-bit compile and runtime
- Support for DB2 10.1
- New DB2 SQL compiler directive option, BGP, to enable background parsing.

Debugging Enhancements

For variables in a copybook that are modified by COPY... REPLACING statements in your code, the Autos window displays all values defined in the source code. When there are multiple COBOL programs in your project that perform a COPY... REPLACING in a copybook, the Autos window only uses the replacing

values found in the first COBOL program. In addition, a new command, **Open copybook with replaced values**, is now available from the editor for the copybooks that the COPY... REPLACING statements modify.

What was New in Visual COBOL 2.1

Visual COBOL 2.1 provided enhancements in the following areas:

- [ACUCOBOL-GT Data Types in Managed Code](#)
- [ACUCOBOL-GT Library Routines in Managed Code](#)
- [Associating file extensions with the COBOL language](#)
- [Compiler Directives](#)
- [.int, .gnt and .lbr File Types Support](#)
- [Managed code enhancements](#)
- [OpenESQL](#)

ACUCOBOL-GT Data Types in Managed Code

ACUCOBOL-GT data types and sign() variants that were previously only available in native code are now supported in managed code. Use the Compiler directives COMP1 and COMP2 to set ACUCOBOL-GT behavior for those particular data types.

ACUCOBOL-GT Library Routines in Managed Code

ACUCOBOL-GT library routines that were previously only available in native code are now supported in managed code.

Associating file extensions with the COBOL language

This release includes enhancements to the way you associate file extensions and extensionless files with the COBOL language.

If you are importing existing COBOL applications into Visual Studio, it is recommended that you create associations with COBOL within the IDE for any extensions that are not traditionally used in COBOL.

Compiler Directives

The following new Compiler directives are now available:

DISPLAY	Defines the default behavior of standard DISPLAY statements.
COMP1	Specifies the behavior of a COMP-1 data item.
COMP2	Specifies the behavior of a COMP-2 data item.
RESTRICT-GOTO	Generates a syntax error for GO TO statements that transfer control to outside of the current section.
ILSMARTRESTRICT	Limits the generation of properties in ILSMARTLINKAGE classes to non-redefining elementary items.

The following Compiler directive has changed:

- **DATAMAP** - Two new parameters allow you to display either the address or offset values for data items in your program.

.int, .gnt and .lbr File Types Support

Support has been added within the IDE for compiling native COBOL applications to the Micro Focus legacy formats .int and .gnt, and to package these files as a Micro Focus library file (.lbr). Improvements include:

- An option to compile all native COBOL projects to `.int` and `.gnt` code. You can set this on the Application page in your project's properties.
- A new native COBOL project template, Micro Focus INT/GNT.
- An option to package the `.int` and `.gnt` files produced by the project as a Micro Focus `.libr` library files.
- Improvements to the Net Express Project Import wizard that enable you to convert existing Net Express projects to Visual COBOL projects that compile to `.int` and `.gnt` code.

Managed code enhancements

Delegates and Events

This release provides support for combining delegates, using the `METHOD` keyword to specify method groups, and implicit conversion from a method group or an anonymous method to the suitable delegate type.

Handling Invalid Numeric Data

The handling of invalid numeric data is controlled by a number of Compiler directives: `HOSTNUMMOVE`, `HOSTNUMCOMPARE` and `SIGNFIXUP`. These directives were previously only available in native code but are now supported in managed code.

Resolving Types

In this release, the Compiler attempts to resolve types to those defined in the current compilation unit wherever possible. The Compiler will attempt to resolve such types to an external name only if no suitable type exists in the current compilation unit. For example:

```
$set ilusing"System"
class-id MyNamespace.EventHandler.
01 o type EventHandler.
end class.
```

In this release, `01 o type EventHandler.` resolves to `MyNamespace.EventHandler` and not to `System.EventHandler`.

Specifying Properties

In previous versions of the products, properties declared using the `PROPERTY` keyword on a data item were generated as final properties. Starting with this release, they are generated as virtual properties by default. In order to make the properties final, you need to specify the word `FINAL` following `PROPERTY`. This change may affect the generation of Proxy classes, for example, if you are using WCF.

OpenESQL

ODBC

Added support for a generic one-phase commit for ODBC XA switch module.

SQL Compiler Directive Options

OpenESQL has been enhanced to support the the following new SQL compiler directive options:

DATE Controls the reformatting of date values in output parameters and in input parameter character host variables when `DETECTDATE` is also specified.

TIME Controls the reformatting of date values in output parameters and in input parameter character host variables when `DETECTDATE` is also used.

DATEDELIM Specifies a single character as the delimiter between the year, month, and day components to override the default delimiter determined by the `HCOSS DIALECT` or `DATE` directive specification.

TIMDELIM Specifies a single character as the delimiter between the hour, minute, and second components to override the default delimiter determined by the `HCOSS DIALECT` or `TIME` directive specification.

TSTAMPSEP Specifies a single character as the separator between the date and time parts of timestamp and date/time data.

OpenESQL Assistant OESQL Assistant now supports updateable cursors.

SQL Server We now support Microsoft SQL Server 2012.

What was New in Visual COBOL 2.0

Visual COBOL 2.0 provided enhancements in the following areas:

- [Building Projects to Multiple Output Files](#)
- [Compiler Directives](#)
- [Debugging Enhancements](#)
- [Library Routines](#)
- [Managed COBOL Language Features](#)
- [Data Access](#)
- [Run-Time Tunables](#)
- [Samples](#)
- [Vision Data File Searching](#)
- [XML Support](#)

Building Projects to Multiple Output Files

You can now build your native Link Library projects to multiple output files. To configure this, you need to go in the project properties and set **Output To** to **Multiple Libraries**.

Compiler Directives

The following new directives are now available:

- **COPYSEARCH** - enables you to specify how copybooks are located. You can choose between usual Micro Focus COBOL behavior or usual RM/COBOL behavior.
- **ILSMARTNEST** - enables you to nest ILSMARTLINKAGE classes inside the program class in which they are defined. This makes it possible to have multiple programs in a single compilation unit that include linkage records with the same name.

The following directives have been changed:

- **DIALECT(RM)** - now accepts a new parameter, **RM**, which enables the RM-compatible functionality that the **RM** directive used to enable.
- **ILUSING** - when set on a single file using the **SET** statement, `$set ilusing`, the directive only affects that file.

Debugging Enhancements

Visual COBOL 2.0 provides the following enhancements to debugging:

Core dump files On Windows, the core dump files created by this product are Microsoft minidump files. If you have a mixed-language application, you can use the minidump file to debug the other languages involved.

Debugging native COBOL Link Library projects You can now debug native COBOL link library projects that get built to a single .dll file.

Program Breakpoints

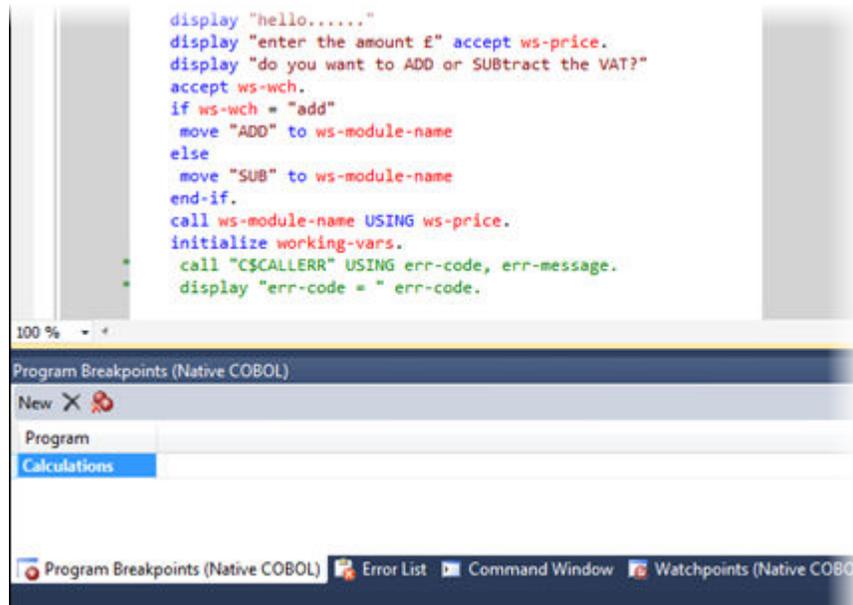


Note: Program breakpoints are supported in native COBOL only, and are not supported with nested programs.

Use the **Program Breakpoints (Native COBOL)** window to add program breakpoints to your applications. When a program breakpoint is set the application breaks whenever the program or one of its entry points is called.

In the **Program Breakpoints (Native COBOL)** window, click **New** and type the name of the source file, without the extension. For multi-program source files, to set a break for a sub-program, use its program-id.

To display the **Program Breakpoints (Native COBOL)** window, click **Debug > Windows > Program Breakpoints (Native COBOL)**.



Step out of OSVS perform statements

You can now step out of a perform statement when PERFORM-TYPE(OSVS) or DIALECT(OSVS) is set.

Support for debug attaching to programs launched by services

Visual COBOL now supports debugging of native CGI applications by waiting for attachment on a directory.

Wait for debuggable attachment option for native COBOL

Wait for debuggable attachment on the Debug page in the properties of native properties has been enhanced. You can now specify one the following options for it - Wait for any program, Wait for directory, and Wait for ID.

Library Routines

The following routine has been enhanced:

- The CBL_SEMAPHORE_ACQUIRE routine now accepts a `timeout` parameter.

Managed COBOL Language Features

The following new syntax elements are now available in managed COBOL:

Local Variables	In managed COBOL, Data items can now be declared in the procedure division, using the DECLARE statement. In addition, they can be declared inline as the iterator in a PERFORM statement, or as an exception message in a TRY ... CATCH ... FINALLY statement block.
Collections	There are two new collection types in managed COBOL: LIST and DICTIONARY. For a LIST, you can add elements to a list, retrieve the nth element of the list, replace the nth element, iterate through the list and clear the list. For a DICTIONARY, you can add key value pairs, retrieve a value corresponding to a key, to replace the value corresponding to a key, iterate through the dictionary and clear the dictionary.
Properties	In managed COBOL, a property can now be defined using PROPERTY-ID and GETTER and SETTER phrases to access to the property. The previous technique of specifying the keyword PROPERTY on a data declaration is still available.
Indexers	In managed COBOL, an indexer can now be defined using INDEXER-ID and GETTER and SETTER phrases to access the indexer value. Indexers are similar to properties, except that their accessors take parameters. Indexers allow instances of a class or valuetype to be indexed just like arrays.
Zero-based Indexing	The managed COBOL syntax for arrays now uses zero-base indexing to access arrays when square brackets are specified. For backward compatibility, one-base indexing is used when round parentheses are specified.

Data Access

Visual COBOL version 2.0 provides the following enhancements:

- Improved IDE integration with SQL directives - now supports handling of deprecated and removed directives. Also supports filtering of the choices offered to the user by product type, project type, and platform.
- OpenESQL has been enhanced and it now:
 - defaults to optimal performance
 - supports 64bit ODBC across all platforms
- The OpenESQL ADO.NET managed run-time has been redesigned so it now works faster. It also supports multiple result sets with stored procedures.

Run-Time Tunables

Visual COBOL 2.0 provides the following new tunables:

- printer_raw_redirection - use this to redirect WRITE statements through the Windows print spooler as RAW data types.
- subsystem_cancel_mode - use this to override the default cancel mode when you use the CBL_SUBSYSTEM library routine to cancel a subsystem.

Samples

The following new samples are now available:

- CGI demonstrations:
 - Complex CGI application - demonstrates how to use native COBOL to create CGI programs which accept data from a form on a Web page and then redisplay that data in another Web page
 - Simple CGI application - demonstrates how to use native COBOL to create CGI programs which accept data from a form on a Web page and then redisplay that data in another Web page
- The following samples have been added to the COBOL for .NET section:
 - Collections - demonstrates the new LIST and DICTIONARY collection types

- Local Variables - shows how to declare data items in the procedure division in the DECLARE, PERFORM and TRY statements
- The code in the Properties sample in the **COBOL for .NET** section of Samples Browser has been enhanced to use the new PROPERTY-ID syntax. The sample also includes a sample program for Indexers which illustrates the new INDEXER-ID syntax.

Vision Data File Searching

Visual COBOL 2.0 provides the following new ACUCOBOL-GT compatible environment variables to help search for Vision data files at run time:

```
APPLY_FILE_PATH
FILE_CASE
FILE_PREFIX
FILE_SUFFIX
```

XML Support

The IBM-style XML GENERATE syntax is now supported in .NET managed COBOL.

What was New in Visual COBOL 2010

Visual COBOL 2010 provided enhancements as part of the following releases:

- [New Features in Enterprise Developer 2010 R4 Update 2](#)
- [New Features in Enterprise Developer 2010 R4](#)

New Features in Visual COBOL 2010 R4 Update 2

Documentation for the Dialog System AddPack

Documentation for the Dialog System AddPack is now available and is integrated with the Visual COBOL documentation.

This documentation describes the AddPack, which enables you to modernize Dialog System applications within Visual COBOL. You migrate an application to Visual COBOL and from there you can run the application without change, or modernize it over time.

The documentation describes some modernization techniques such as:

- A Windows Forms form replacing a Dialog System dialog, where the form can contain .NET controls. See the Customer + .NET WinForm sample `CustomerWinForm.sln`.
- A Windows Forms control wrapped as an ActiveX control and used on a Dialog System dialog. See the Customer + .NET GridView User Control sample `custgrid.sln`.
- A WPF user control hosted by a Windows Forms user control, which is then exposed as ActiveX ready for use by Dialog System. See the Customer + .NET WPF GridView User Control sample `CustGridWPF.sln`
- A .NET managed code application interacting with Dialog System as native COBOL .dll. See the Managed Customer sample `ManagedCustomer.sln`.

OO COBOL Class Library Reference

Help for the following OO COBOL class libraries is available:

```
Base class library
GUI class library
OLE class library
```

OLE Automation class library

The Help is available in the file `nrxclr.chm`, which is installed in the Help folder of your installation. The default location is `%ProgramFiles(x86)%\Micro Focus\Visual COBOL\Help`.

To open the help, double-click `nrxclr.chm` in Windows Explorer.

Net Express Project Import Wizard

Visual COBOL for Visual Studio includes a Net Express Project Import Wizard that converts NetExpress projects into Visual Studio solutions. The wizard analyzes the Net Express project file and its configuration settings, creates one or more Visual Studio native projects based on this information, imports the existing source code into them and sets the appropriate directives. When the process is complete, it can optionally display a detailed conversion report.

Features Added in Visual COBOL 2010 R4

ACUCOBOL-GT Compatibility

The Compiler and run-time continue to provide support for ACUCOBOL-GT. The directive `ACU` is the main switch for turning on ACUCOBOL-GT compatibility. The `ACU` directive enables various ACUCOBOL-GT syntax extensions and other language elements. Additional ACUCOBOL-GT compatibility features include the following:

- When using a `CALL` statement, the `USING` and `GIVING/RETURNING` phrases can now appear in either order.
- The following ACUCOBOL-GT standard library routines can now be used with Visual COBOL in native code:
 - `C$CALLED`
 - `C$CALLED`
 - `C$CHDIR`
 - `C$MAKEDIR`
 - `C$MEMCPY`
 - `C$MYFILE`
 - `C$PARAMSIZE`
 - `C$RERR`
 - `M$ALLOC`
 - `M$FREE`
 - `M$COPY`
 - `M$FILL`
 - `M$GET`
 - `M$PUT`
 - `WIN$VERSION`
- The following ACUCOBOL-GT 'ccbl' compiler options can now be used with Visual COBOL:
 - `-E`, `-V`
 - `-Cv`
 - `-Da`, `-Db`, `-Dd31`, `-DL1/2/4/8`, `-Dq`, `-FpRounding`
 - `-La`, `-Li`, `-Lc`, `-Lf`, `-Ll`, `-Lo`, `-Ls`, `-Lw`

Note: The output that these list options provide differs in Visual COBOL.

 - `-Qm`
 - `-Rc`, `-Rn`, `-Rw`
 - `-Sa`, `-St`, `-Sd`, `-Sp`, `-S1...-S9`
 - `-noTRUNC`, `-truncANSI`, `-Dz`

- -Td, -Te
- -Vc
- -Za, -Zc, -Zl, -Zn, -Zs, -Zi, -Zr1, -Zy, -arithmeticVSC2

Full ACUCOBOL-GT compatibility is documented under the *Programming* section in the product help.

ADO.NET Runtime Improvements

The OpenESQL ADO.NET runtime has been re-engineered, offering several advantages over previous versions, providing:

- Better performance - the new runtime is much faster and can as much as double your performance, depending on the specifics of your application.
- Support for multiple run unit processing.
- Support for both client applications and SQL CLR stored procedures.
- For stored procedure applications:
 - Support for input/output parameters defined as COBOL data types, in addition to the previously supported .NET data types
 - Support that enables stored procedures to return more than one result set.
 - Static SQL Support for the DIALECT=MAINFRAME Compiler directive.
- For client-based applications:
 - Support for dynamic SQL that uses an SQLDA structure to pass parameter and result host variables, specifically including the following embedded SQL statements:

- PREPARE - For example, the following statement passes a prepared statement into the SQLDA from a host variable:

```
EXEC SQL
    PREPARE stmt1 INTO :sqlda FROM :stmt-buf
END-EXEC
```

- EXECUTE - For example, the following statement executes a prepared statement using a previously constructed SQLDA structure:

```
EXEC SQL
    EXECUTE stmt1 USING DESCRIPTOR :sqlda-structure
END-EXEC
```

- OPEN - For example, the following statement opens a cursor using a previously constructed SQLDA structure:

```
EXEC SQL
    OPEN c1 USING DESCRIPTOR :sqlda-structure
END-EXEC
```

For complete information on using these statements, see the *PREPARE*, *EXECUTE*, and *OPEN* topics.

- Additional support for result set processing that allows you to receive more than one result set back from a stored procedure.
- Support for applications built to target .NET Framework versions 2.0, 3.0 and 3.5, in addition to .NET Framework version 4.0.



Important: To take advantage of these improvements you must recompile your applications with the DBMAN=ADO Compiler directive. However, if you want to continue using the functionality available in previous releases, we now provide the DBMAN=OADO directive to use in place of the DBMAN=ADO directive. DBMAN=OADO provides the same functionality as was available with previous versions of the DBMAN=ADO Compiler directive, and has 100% backward compatibility with applications designed and developed with Visual COBOL R3.

Creating Projects from Selected Files

A new option, Create Project From Selection, is now available for your projects in Solution Explorer. You can select a number of COBOL files and copybooks in your project and opt to create a new project from them in the same solution.

Debugging Enhancements

The ability to load core dump files in Visual Studio has been added. This feature works with native COBOL only.

Documentation

If you are using Visual Studio 2010 Service Pack 1, the help is displayed in a stand-alone help viewer with an index and a fully expandable table of contents.

Embedded HTML

We now support the use of Embedded HTML (EHTML) in COBOL CGI programs, which enables you to output HTML directly from your applications.

Improvements to the Implements Smart Tag

The implements smart tag now supports value-types in addition to classes.

Language Improvements

The following improvements have been made to managed COBOL:

- | | |
|--|--|
| Extension methods and extending operators | Managed COBOL now supports extension methods. This feature enables you to add methods to existing types without the need to edit or recompile the code. You can also extend operators. |
| The SYNC modifier for methods | The SYNC modifier locks the values of the arguments sent to the method, so that they do not change while the method is processing. |
| Nested classes | In managed COBOL, a nested class can now be defined so that it can access the instance fields, properties and methods in its containing class. To allow this, you add the optional SHARING PARENT phrase to the nested class definition. |

Large Projects Support

Visual COBOL has been optimized to work with bigger, more complex applications. This includes faster processing of multiple files and various IDE features that facilitate the process of developing large-scale project.

You can quickly move existing COBOL code into Visual Studio with the help of various wizards and windows such as the Create Project from Existing Code wizard and the Create Project from Selection wizard. The IDE is preconfigured so that during the file import it automatically scans the files and sets Compiler directives on them as appropriate.

New Compiler Directives

The following new Compiler directives are provided:

- ILCUTPREFIX - removes a specified prefix from the names of the COBOL data items in your source code.
- ILSMARTLINKAGE - exposes the Linkage Section and entry points to managed code by creating types.
- RUNTIME-ENCODING - determines the runtime encoding.
- SOURCE-ENCODING - passes the encoding of the source program to the Compiler.

New Samples and Tutorial

New samples and a new tutorial showing how to create WCF services in COBOL are available.

Project Details Window

A new window, Project Details, is available for your COBOL projects and solutions showing a complete list of the files in a project or a solution and various file details. You can open the window from the context menu for a project or a solution in Solution Explorer.

Project Properties Updates

The project properties pages have been restructured to make setting options more intuitive.

RM/COBOL Compatibility

The Compiler and run-time continue to provide support for RM/COBOL. Additional RM/COBOL compatibility features include the following:

- The following RM/COBOL standard library routines can now be used with Visual COBOL in native code:
 - C\$Century
 - C\$ConvertAnsiToOem
 - C\$ConvertOemToAnsi
 - C\$DARG
 - C\$Delay
 - C\$GetEnv
 - C\$GetNativeCharset
 - C\$LogicalAnd
 - C\$LogicalComplement
 - C\$LogicalOr
 - C\$LogicalShiftLeft
 - C\$LogicalShiftRight
 - C\$LogicalXor
 - C\$NARG
 - C\$SetEnv
 - C\$RERR
 - DELETE
 - RENAME
- The RM/COBOL file handler can now be used with Visual COBOL, enabled by using the CALLFH(ACUFH) Compiler directive, and then configuring an add-on to the Vision file handler.

Full RM/COBOL compatibility is documented under the *Programming* section in the product help.

Smart Linkage

Exposing COBOL group items as managed types

You can expose COBOL Linkage sections to other managed languages by using the ILSMARTLINKAGE directive. Smart Linkage saves the need to edit your original COBOL code or write wrapper classes.

WCF Services and Service References

Support is now available for adding WCF services as service references to your COBOL projects.

Note: WCF is not supported in the Visual Studio Shell but adding service references for client applications is supported.

XML Extensions



Note: This functionality is supported in native COBOL only.

You can now use XML Extensions, the system that enables your COBOL applications to interact with XML documents, with Visual COBOL.

XML Extensions has many capabilities. The major features support the ability to import and export XML documents to and from COBOL working storage. Specifically, XML Extensions allows data to be imported from an XML document by converting data elements (as necessary) and storing the results into a matching COBOL data structure. Similarly, data is exported from a COBOL data structure by converting the COBOL data elements (as necessary) and storing the results in an XML document.

For more information about XML Extensions, refer to the *XML Extensions User's Guide*, available from the RM/COBOL product documentation set, in the SupportLine section of the Micro Focus Web site.

Features Added in Visual COBOL 2010 R3

.NET COBOL Syntax Improvements

Quoteless syntax	Quotes are not needed when defining types, classes or methods, or when invoking classes and methods.
Construct improvements	The structure of class-id, method-id, enum-id, delegate-id, interface-id, valuetype-id has been improved.
Environment division, Configuration section, Repository	The Environment division, Configuration section and the Repository are no longer needed.
Static, Factory and Object blocks	The Static, Factory and Object blocks are no longer needed.
Attributes, Custom-Attribute and Class-Attributes	CUSTOM-ATTRIBUTE is now replaced by the ATTRIBUTES phrase. You no longer need to define class-attributes. Instead, specify the class custom attributes in the class definition.



Tip:

- Visual COBOL supports the older syntax, so projects that are using it will still compile. However, it is recommended to create applications using the new syntax and adhere to the *.NET COBOL Best Practices*.
- It is recommended to use the COBOL project and file templates, snippets and Intellisense as they use the new syntax. To see the new syntax in action, check the Visual COBOL samples.

The following is a more detailed overview of the changes in the syntax with examples:

Quoteless Syntax

Quotes are no longer needed when you define types, classes or methods, or when you invoke classes and methods. For example:

New Syntax	Old Syntax
<code>01 01 type MyClass</code>	<code>01 01 type "MyClass"</code>
<code>type MyClass::New</code>	<code>type "MyClass"::"New"</code>
<code>set o to new MyClass</code>	<code>set o to new "MyClass"</code>
<code>set class::Property to value</code>	<code>set "class"::"Property" to value</code>

New Syntax	Old Syntax
set return-value to class::Method(param1)	set return-value to "class"::"Method"(param1)
invoke class::Method(param1)	invoke "class"::"Method"(param1)
<pre> method-id MyMethod public. local-storage section. procedure division. goback. end method. </pre>	<pre> method-id. "MyMethod" public. local-storage section. procedure division. goback. end method "MyMethod". </pre>

Construct of class-id, method-id, enum-id, delegate-id, interface-id, valuetype-id

The construct of class-id, method-id, enum-id, delegate-id, interface-id, valuetype-id has been improved as follows:

- You do not have to type a period after the declaration (for example, method-id *MethodName*).
- Quotes are no longer required around names.
- You do not need to use the name in the end marker.

New Syntax	Old Syntax
<pre> class-id Namespace.MyClass. object-storage section. method-id InstanceMethod. local-storage section. procedure division. goback. end method. static. method-id StaticMethod public local-storage section. procedure division. goback. end method. end class. </pre>	<pre> class-id. MyClass as "Namespace.MyClass". environment division. configuration section. repository. static. working-storage section. end static. object. working-storage section. method-id. "InstanceMethod". local-storage section. procedure division. goback. end method "InstanceMethod". end object. end class MyClass. </pre>

Environment Division, Configuration Section, Repository

You no longer need to use an Environment division, a Configuration section or a Repository paragraph. For example:

New Syntax	Old Syntax
<pre> program-id. Program1 as "MyProject.Program1". </pre>	<pre> program-id. Program1 as "MyProject.Program1". </pre>

New Syntax	Old Syntax
<pre> data division. working-storage section. procedure division. goback. end program Program1. </pre>	<pre> environment division. configuration section. repository. data division. working-storage section. procedure division. goback. end program Program1. </pre>

Static, Factory and Object Blocks

The Static and Object blocks are no longer used. With the new syntax you need only one working-storage section for items that were defined in a static or object block under the old syntax.

To define a static method, use the STATIC word.

The following example shows how to define static methods with the new syntax and how to avoid using an object block:

New Syntax	Old Syntax
<pre> class-id Namespace.MyClass. working-storage section. 01 my-object-data pic x. 01 my-static-data pic x static. method-id InstanceMethod. local-storage section. procedure division. goback. end method. method-id StaticMethod public static. local-storage section. procedure division. goback. end method. end class. </pre>	<pre> class-id. MyClass as "Namespace.MyClass". environment division. configuration section. repository. static. working-storage section. 01 my-static-data pic x. end static. object. working-storage section. 01 my-object-data pic x. method-id. "InstanceMethod". local-storage section. procedure division. goback. end method "InstanceMethod". end object. end class MyClass. </pre>

Attributes, Custom-Attribute and Class-Attributes

These are the changes for CUSTOM-ATTRIBUTE and class-attributes:

- The CUSTOM-ATTRIBUTE phrase is replaced by the ATTRIBUTE phrase.
- You no longer have to define class-attributes. Instead, specify the class custom-attributes in the class definition using the ATTRIBUTE phrase.
- Quotes are no longer needed around the name of the attribute and you can omit the word "Attribute" from the name.

For example:

New Syntax	Old Syntax
<pre> class-id MyNamespace.MyClass. attribute Serializable. working-storage section. ... end class. </pre>	<pre> class-id. MyClass as "MyNamespace.MyClass" . class-attributes. custom-attribute is type "SerializableAttribute" . object. working-storage section. ... end object. end class MyClass. </pre>

Creating Projects from Existing Code

Now you can create Visual Studio COBOL projects from existing applications using the Create New Project from Existing Code Files wizard. The wizard will create a new COBOL project and automatically add files to it from the specified directories. It will perform an automatic file scanning to identify which files are programs and copybooks, so that they can be correctly added to the project.

Create New Project from Existing Code Files

Specify Project Location, Project Type and Source Files
You can choose the files from one or more folders

Project file location:

Project name:

Project type:

Add files to the project from these folders

Folders:

Add subfolders	Folder

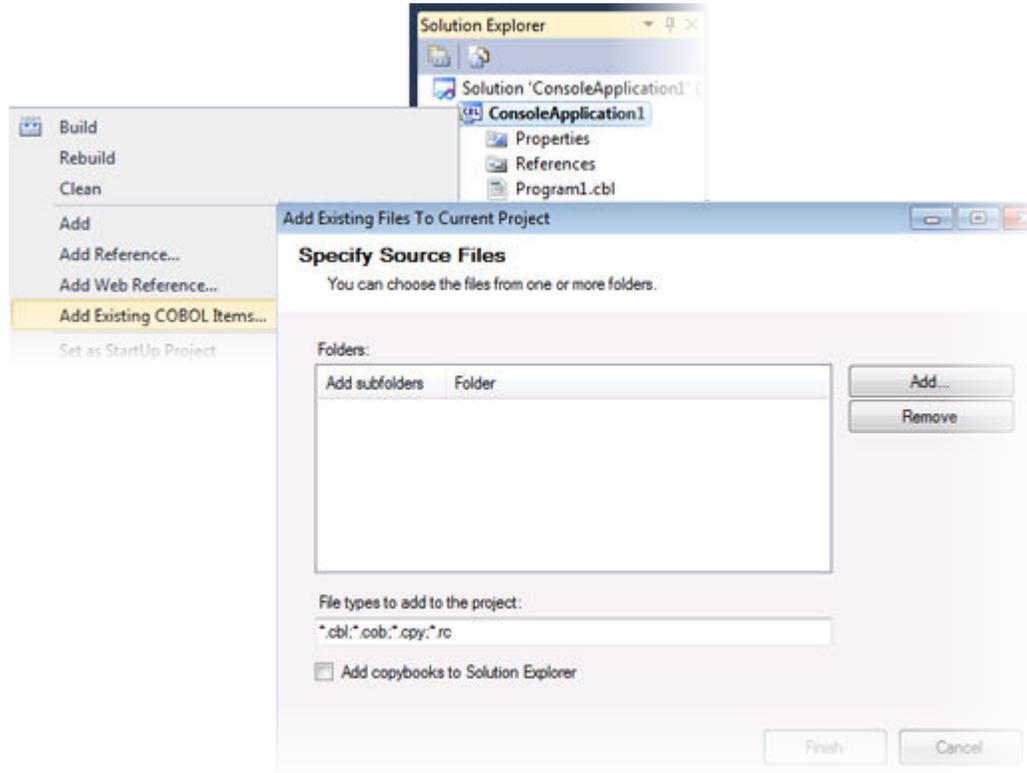
File types to add to the project:

Add copybooks to Solution Explorer

< Previous Next > Finish Cancel

Add Existing COBOL Items Wizard

You can add existing COBOL files to your Visual Studio project using the new Add Existing COBOL Items wizard available from the context menu of the project in Solution Explorer. The COBOL files will be scanned to determine which ones are programs or copybooks, and then they will be added to the project.

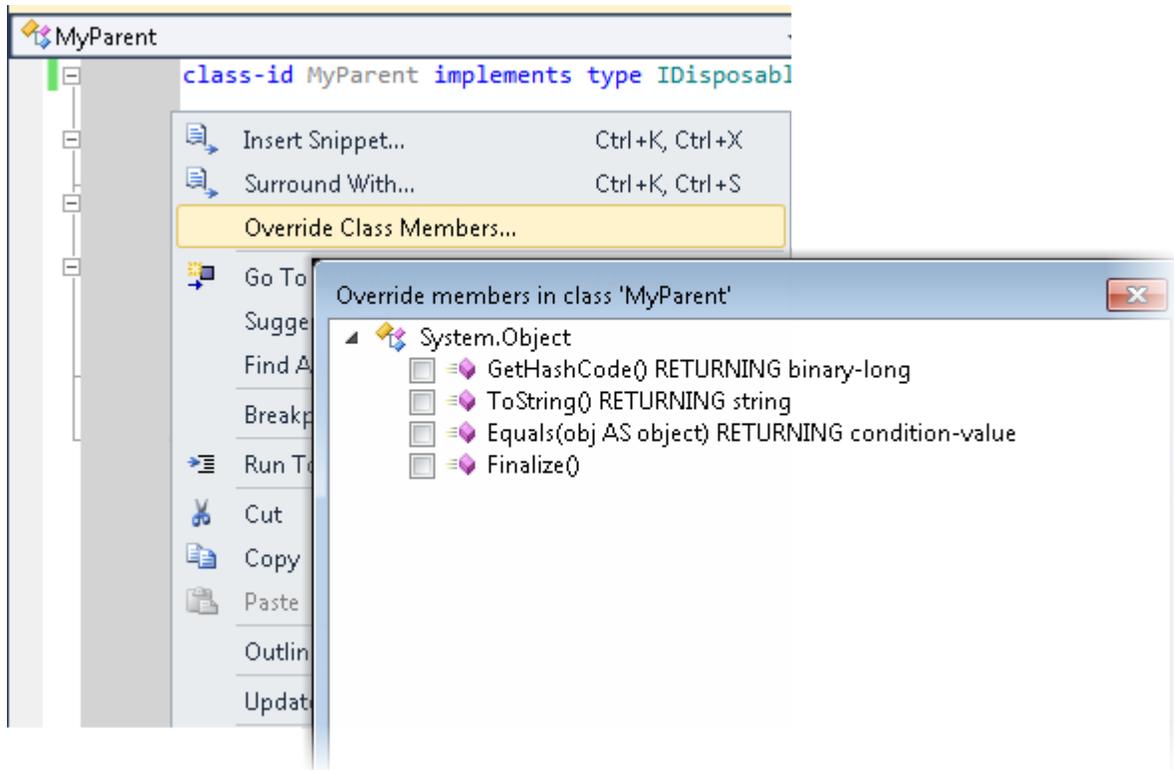


Override Class Members Dialog



Note: This feature works with .NET managed code only.

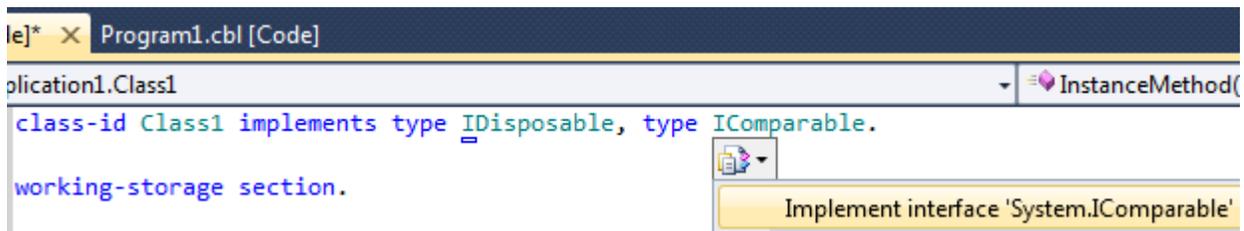
The new Override Class Members dialog available in the editor enables you to override the members of inherited classes. The dialog helps you see the base classes from which a class inherits, select the members to override and add the construct of the overriding methods to the class.



Smart Tag for Implementing Interfaces

 **Note:** This feature works with .NET managed code only.

You can now easily implement interfaces with the help of a Smart Tag. The tag appears underneath at the beginning of the declaration of any interface that is not fully implemented. To implement the interface, you simply need to click the tag.



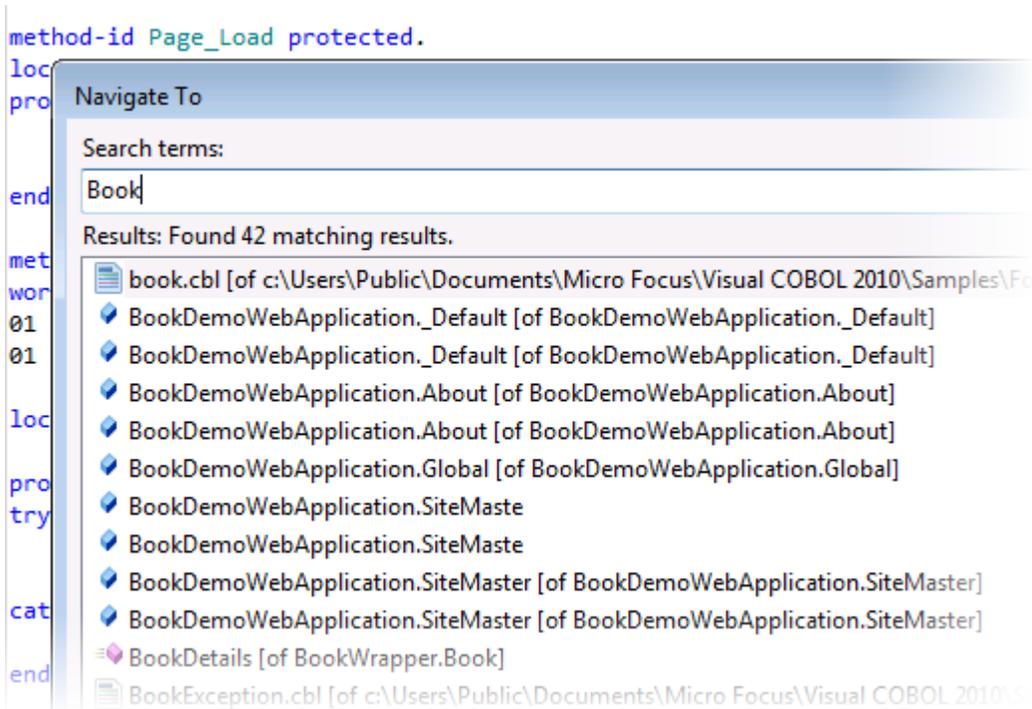
Snippet for Implements

 **Note:** This feature works with .NET managed code only.

The snippet for implements has been improved. It now automatically implements the members from an interface and has improved support for more complex method signatures.

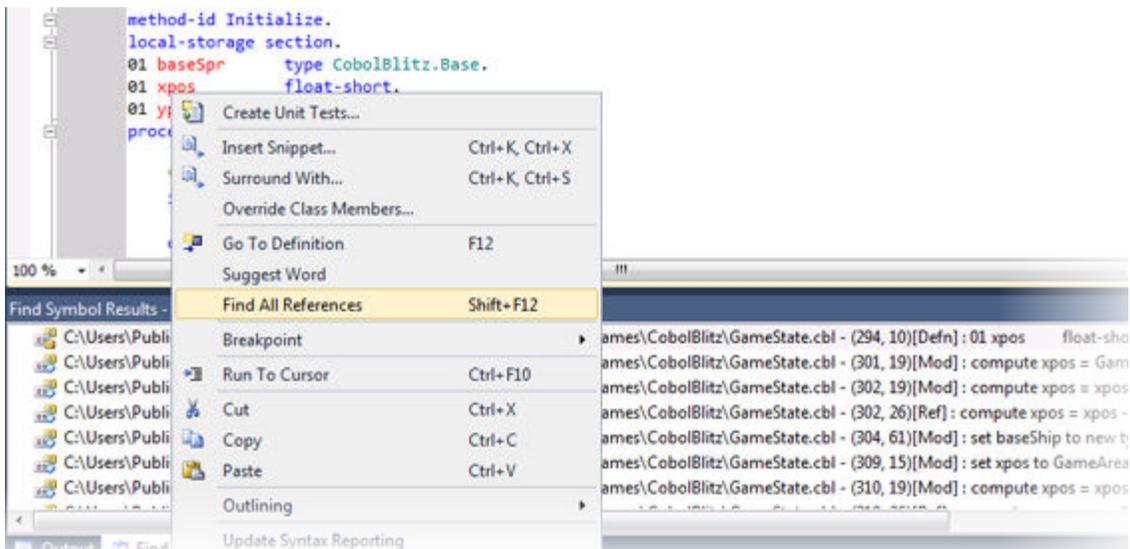
Navigate To

Use the Navigate To option in the Edit menu to search for files, variables and section names in all projects and files in your solution.



Find All References

The Find All References option available from the editor enables you to search for references of COBOL data items, section or paragraph names in your solution.



Web Application Projects

 **Note:** This feature works with .NET managed code only.

This release offers Web Application Project templates for creating COBOL Web applications and Web sites and applications. The benefits of using a Web Application project include:

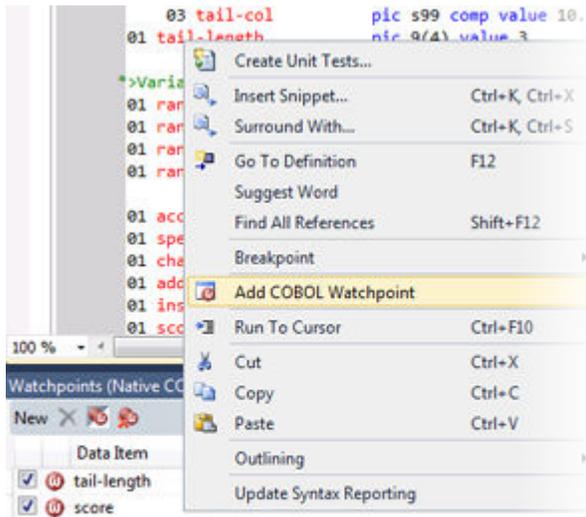
- A Web Application project includes a project file which enables you to specify what files are part of the project and should be compiled.
- It adds namespaces for all items of the project.

- The source code is compiled into a single assembly on your local machine and is then deployed to the IIS server. You don't have to deploy the code behind.
- A Web Application project includes a "Publish" option for deploying the compiled assembly to an IIS server directly from the IDE using the automated tools of Visual Studio.
- Supports the Visual Studio Code Analysis feature.

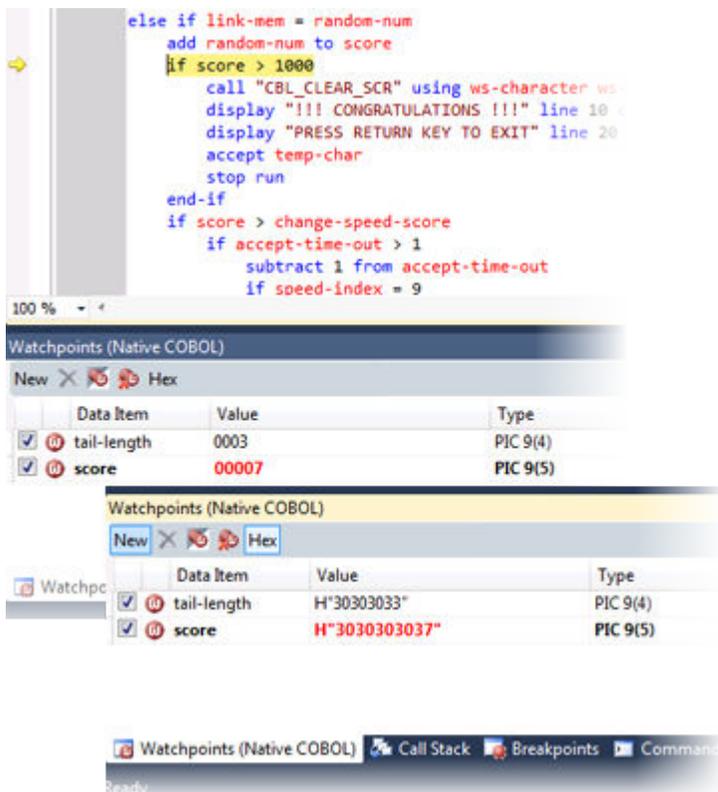
Debugging

The following debugging enhancements have been made:

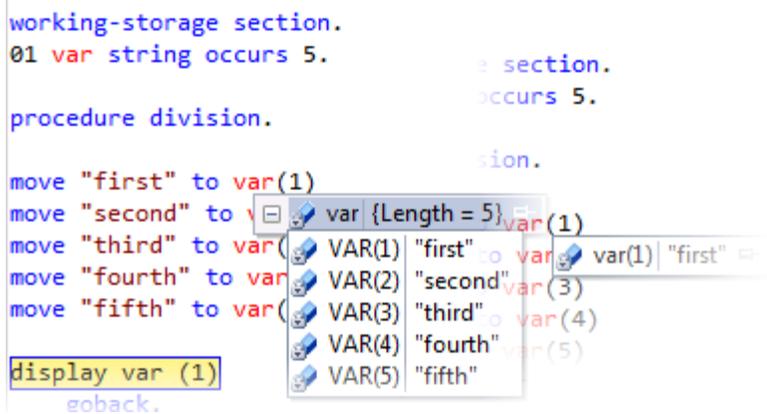
- COBOL watchpoints and break on data change - you can set COBOL watchpoints on individual data items in native COBOL. COBOL watchpoints enable you to watch the area of memory associated with the particular data item and help track memory corruption. When the memory changes, debugging stops on the line that immediately follows the line on which the data has changed. This feature works with native code only.



- **Watchpoints (Native COBOL)** window - enables you to manage the COBOL watchpoints you add to your applications and view the contents of the memory associated with each watchpoint. This feature works with native code only.

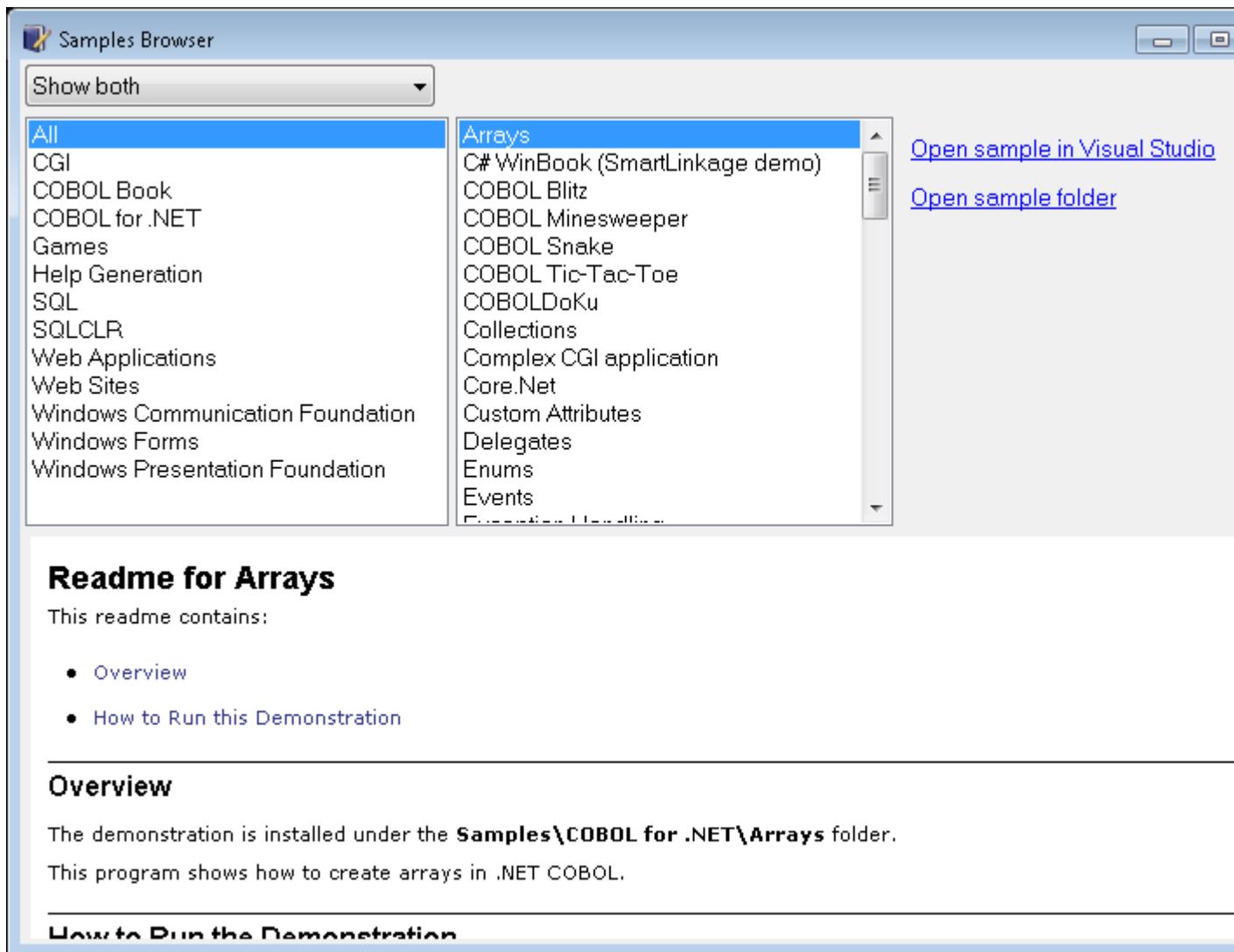


- Simplified remote debugging - a simplified process for setting up remote debugging is provided.
- Attach to 64-bit process and debug - provides the ability to attach to and debug 64-bit COBOL processes.
- Debug tooltip for OCCURS items - you can now specify whether the debug tooltips for OCCURS items should display all items in an array or the value of an expression.



Samples Browser

You can preview the samples and access them more easily with the help of the Samples Browser which is now available from the Start menu. Samples Browser lists the samples by category



SQL CLR Integration Support

The new SQL CLR Integration support enables you to create stored procedures, user-defined functions and types, aggregates, and triggers in managed code, taking advantage of the Microsoft SQL Server 2008 R2 SQL CLR feature.

ACUCOBOL-GT Compatibility and RM/COBOL

The Compiler and run-time now include initial support for ACUCOBOL-GT. This support is enabled by several new Compiler directives. The directive ACU is the main switch for turning on ACUCOBOL-GT compatibility. The ACU directive enables various ACUCOBOL-GT syntax extensions and other language elements. Additional ACUCOBOL-GT compatibility features include the following:

- Vision indexed file system and utilities (vutil, vio, and logutil) support. Vision support is enabled by the new CALLFH(ACUFH) option.
- ACUCOBOL-GT compiler options. By using the new ACUOPT directive you can specify the same options available in ACUCOBOL-GT.
- ACUCOBOL-GT and RM/COBOL data type support. This enables you to mix ACUCOBOL-GT and Micro Focus Visual COBOL applications via data files or calls.
- Initial ACUCOBOL-GT syntax support. The Compiler now supports some of the ACUCOBOL-GT extensions.

- Interoperability between ACUCOBOL-GT and Visual COBOL components. You can begin to build applications that combine ACUCOBOL-GT with Visual COBOL features.

ACUCOBOL-GT compatibility is documented under the *Programming* section in the product help.

XML Support

Enhancements have been made to XML Parse/Generate to provide compatibility with IBM® Enterprise COBOL for z/OS® v4.2.

New in XMLGENERATE:

- ATTRIBUTES phrase
- NAMESPACE and NAMESPACE-PREFIX phrases
- XML-DECLARATION phrase

New in XMLPARSE:

There are now two modes of XMLPARSE support using the XMLPARSE() compiler directive:

- XMLPARSE(COMPAT) – provides compatibility with IBM Enterprise COBOL for z/OS v4.1 and earlier.
- XMLPARSE(XMLSS) – provides compatibility with IBM Enterprise COBOL for z/OS v4.2.

XMLPARSE(XMLSS) provides:

- ENCODING phrase
- RETURNING NATIONAL phrase
- VALIDATING phrase
- New special registers - XML-NAMESPACE, XML-NNAMESPACE, XML-NAMEPSACE-PREFIX and XML-NNAMESPACE-PREFIX.
- New behaviors – for example, different return codes, different output registers depending on the EVENT.

Note: The ability to parse XML documents one segment at a time with the help of the END-OF-INPUT XML event is not supported yet.

Features Added in Visual COBOL 2010 R2

File Handler

The Micro Focus File Handler is now provided as both verifiable and non-verifiable versions. Compiling your application with the ILVERIFY directive will automatically reference the verifiable File Handler assembly.

Go To Procedure Division

The Go To Procedure Division button is now available on the Go To Location toolbar. Clicking the button positions the cursor on a Procedure Division depending on the current context of the code.

OpenESQL Assistant

Support for the OpenESQL Assistant has been added. The OpenESQL Assistant is an interactive tool that enables you to easily design and build SQL queries and embed those queries into your COBOL code.

Features include:

- Prototype SQL SELECT statements and test them against a database
- Design SQL INSERT, UPDATE, and DELETE statements
- Insert SQL queries into the COBOL code
- Create and insert auxiliary code into your COBOL code

Samples

The following games have been added to the samples:

- COBOL Blitz - A shooter game in which the players use a laser cannon to defend themselves against the invasion of aliens troops. The goal is to destroy the troops and prevent them from reaching the bottom of the screen.

Special Features:

- 2D graphics
- Audio effects
- Snake - An arcade game in which the player navigates a long chain of symbols across the screen and scores by collecting numbers. Numbers add to the overall length and the speed of the snake. The player needs to avoid hitting the borders of the screen or touching the snake's body as this terminates the game.
- Tic-Tac-Toe - The player competes with the PC to place three identical marks in a horizontal, vertical, or diagonal row on the 3x3 board.

Snippets

This release provides new snippets for Attribute, DateTime, Implements and for static methods.

SQL Support

The SQL technology that was present in Net Express is now seamlessly integrated within the Visual Studio 2010 development environment. When you develop COBOL SQL applications in Visual Studio, you can use the same development environment to extend and modernize your COBOL assets.

Features include:

- OpenESQL technology that supports embedded SQL in your COBOL applications
- OpenESQL Assistant that automatically generates embedded SQL in a COBOL program template given basic database information
- DB/2 ECM technology that uses embedded SQL to work with DB2 LUW
- COBSQL processor that provides native DBMS SQL support for Oracle's Pro*COBOL and other vendors

XML Parse/Generate

Visual COBOL now supports the IBM-style XML syntax and enables your applications to process XML data. Support for the XML PARSE and XML GENERATE statements is provided in the Visual COBOL compiler.

Features Added in Visual COBOL 2010 R1

Visual Studio as the Core Integrated Development Environment

The Visual Studio editor has been extended in a number of ways to enhance its support for COBOL, including Standard Visual Studio 2010 features for program navigation are exploited for COBOL applications.

- Fully integrated COBOL development environment delivers high programmer productivity by exploiting Visual Studio tools and providing instant feedback.
- Enhanced COBOL syntax for .NET programmers makes it easier for COBOL programmers to use .NET services or for programmers with .NET experience in other programming languages to be productive with COBOL.
- Visual COBOL supports the development and deployment of both "managed" .NET (with multi-targeting for .NET Framework V4 and earlier versions) and "unmanaged", native code applications.

- Visual COBOL is a part of the Visual Studio 2010 product portfolio from Micro Focus which also includes testing and developer productivity tools.
- COBOL Margins - visual indication of COBOL margins which are sensitive to the COBOL margin directive currently selected for the program – if the setting is changed via an embedded "\$SET SOURCEFORMAT" directive, then the display is immediately updated.
- COBOL sensitivity is extended to support COBOL methods and data items in IntelliSense and preconfigured "code snippets" reduce the effort required to complete code and avoid errors being introduced.
- Background parsing continuously ensures that the code being worked on will compile cleanly.

COBOL 2010

Visual COBOL supports the development of both "managed" code which is fully interoperable with other .NET languages and "native code". It is built on a new Micro Focus COBOL platform "COBOL 2010".

A standalone COBOL Server is available for deploying applications developed within Visual COBOL.

Visual COBOL provides a test license version of the COBOL Server to allow system testing before deployment into production.

COBOL Language Extensions

Historically, COBOL has been case-insensitive which makes interoperation with .NET methods more difficult than it should be. For example, method or member names had to be enclosed within quotation marks and declare synonyms to refer to external types. With Visual COBOL these restrictions have been removed and the code is more ".NET-like" while still retaining COBOL's traditional ease of understanding. Unnecessary COBOL elements such as "repository" have been made optional which greatly reduces the size and complexity of a COBOL .NET program. The language changes improve readability and simplify the learning process for existing C# or VB programmers who can easily work on the COBOL code. With this flexibility, teams can be more agile and thus reduce development and maintenance costs.

Significant Changes

This section describes significant changes in behavior or usage in each successive release. These changes could potentially affect the behavior of existing applications or impact the way the tools are used.

Where present, the numbers that follow each issue are the Support Incident Numbers followed by the Reported Problem Incident (RPI) number (in parentheses).

Significant Changes in Visual COBOL 4.0

This section describes significant changes in behavior or usage. These changes could potentially affect the behavior of existing applications or impact the way the tools are used.

Where present, the numbers that follow each issue are the Support Incident Numbers followed by the Reported Problem Incident (RPI) number (in parentheses).

- [Code Coverage](#)
- [Codeset Support](#)
- [Common Communications Interface](#)
- [Communications Server](#)
- [Compiler](#)
- [Data Tools](#)
- [Documentation](#)
- [Enterprise Server](#)
- [Enterprise Server Auditing](#)

- [Executables require relinking](#)
- [File Handling](#)
- [MF Server Administrator \(GUI\)](#)
- [Micro Focus Directory Server](#)
- [Run-Time System](#)

Code Coverage

[Back to the list](#)

- Schema changes that affect the test coverage results generated from the `tcutil` utility mean that if you propagate the results to a third-party application (for example, an XSLT processor), and rely on the `<copyFileCoverage>` element, you need to alter your transformations to focus on `<sourceFileCoverage>` instead. The element was renamed to more appropriately reflect its contents, as `tcutil` now gives global coverage for all source files (not just copybooks).

Codeset Support

[Back to the list](#)

- Code-set mappings between ASCII and EBCDIC have been updated when Simplified Chinese is the language in effect. ASCII table 5210 now maps to EBCDIC CCSID 836 for SBCS conversions. This replaces the previously conversion (where ASCII table 1042 was used), which would convert the “\” character to “\$”.
3124321 (1111464)
- New single-byte character set tables for MF CODESET have been added in order to improve support for DB2 LUW - both for off-mainframe databases and for access to z/OS DB2. A number of existing MF CODESET mappings have also been updated. See 'Supported Country Codes' for a full listing of ASCII/ANSI <-> EBCDIC mappings.
3111843 (1109984)

Common Communications Interface

[Back to the list](#)

- You can now configure the Micro Focus Directory Server and enterprise server region's listeners to only use the server's configured SSL and TLS protocols and define a priority ordered cipher suite collection. This forces connecting clients to use the server's preferred ordered list of cipher suites when using the specified protocols.
2866265 (1105526)
- In some circumstances it was possible for a connection to incorrectly accept the identity of an SSL/TLS peer and allow a connection to complete when the connection should have been denied. This occurred due to a failure to check the peer's entire identity certificate chain. This has now been fixed. NOTE: You might need to correct your system's configured certificate chains that fail verification checks at secure connection creation time.
- In some circumstances it was possible to crash the CCITCP module when it was attempting to obtain detailed error information about a closed connection. This has been fixed.

Communications Server

[Back to the list](#)

- TN3270 conversations to Enterprise Server now correctly handle the receive (idle) timeout setting configured for the listener. There are also two new settings for configuring TN3270 timeouts, "Printers time out" and "Output resets timeout". See the online product documentation for more information.
3144133 (1113024)

- MFCS listeners can now be SSL-enabled without the need to have DemoCA installed.
2868627 (1105777)
- MFCS no longer initializes the Security Facility if there are no External Security Managers defined for the region.

Compiler

[Back to the list](#)

- Programs containing EVALUATE statements of the form: EVALUATE true | false WHEN conditional-expression where conditional-expression included inline method invokes would give an RTS 114 error when run as .int code, and an "Illegal .int code" error when generated. This has been fixed. Also, short circuit evaluation is now correctly observed, such that when evaluating condition-1 AND condition-2, if condition-2 contains an inline method invoke and condition-1 is false, then the inline method invoke is not executed. Previously, despite being correctly evaluated, the inline method invoke in condition-2 was being executed. Similar behavior relating to OR evaluations has also been corrected.
3138510 (1112492)
- During compilation, characters within literals that are unknown in the current locale are now less likely to cause spurious errors. However, the correct (and safest) solution is to ensure that the locale has been set correctly, to match the source encoding of these characters. On UNIX, this means setting LANG, LC_CTYPE, or LC_ALL appropriately; each of these variables takes precedence over the former. On Windows, this means setting the system locale in the 'Region and Language' section of Control Panel.
3123935 (1111148)
- The Compiler now produces an E level message - COBCH1888 Typedef is defined differently in another external program - if different external programs have conflicting definitions of the same typedef name. (To restore the previous behavior, where the earlier definition was ignored, use the directive HIDEMESSAGE"1888".)
- An issue with the Compiler has been fixed so that in the RECORD VARYING clause, if the minimum and maximum lengths are specified, the maximum length must be greater than the minimum length.

Data Tools

[Back to the list](#)

- Records with an invalid value for a conditional field will no longer result in a match for that conditional layout.
2853226 (1103406)

Documentation

[Back to the list](#)

- You can use 'byte' or 'BYTE' as a synonym for the binary-char unsigned data type. As a result, 'byte' and 'BYTE' are now reserved words in Managed COBOL. Use the REMOVE"BYTE" Compiler directive to prevent an error being produced for existing programs that use the reserved word as a user-defined word.
3147576 (1113323)
- As of version 3.0, references to types within an assembly other than mscorlib need to be explicitly referenced. You can achieve this by using the ILREF Compiler directive. (Previously, in certain circumstances, the Compiler would allow access to types within the System.dll assembly without the need for an ILREF"System" directive.
3121002 (1111373)

Enterprise Server

[Back to the list](#)

- The External Security Facility (ESF) can now be configured to throttle large volumes of incoming Verify (user authentication / signon) requests to improve resilience to denial-of-service and brute force attacks. See "Verify Request Throttling" for more information.

3113639 (1110160)

- The LDIF files used to create the sample configuration for Enterprise Server LDAP-based security no longer create an empty "PHYSFILE" resource class. Changes in the JCL engine as of ES 3.0 caused most jobs to fail when submitted to a security-enabled region using such a configuration. See the product help for more information.
- The MLDAP ESM Module, part of the Enterprise Server External Security Facility, now supports the Argon2 hash algorithm for creating password verifiers. See MLDAP ESM Module Custom Configuration Information in the product help for more information. NOTE: The Argon2 hash is optional and not enabled by default.
- The MQ pages in ESMAC are now controlled by a new security resource, MQL. This enables you to either restrict or grant users access. 'MQL*' is a new resource that needs to be added under MFESMAC similar to existing resources such as 'PCT*' or 'XAT*'. The following is a sample export of the LDAP repository:

```
*****
# Sample security definitions for ESMAC MQ Listeners/Writers pages

#####
##### MQL*          ##
#####
dn: CN=MQL*,CN=MFESMAC,CN=Enterprise Server Resources,CN=Micro
Focus,CN=Program Data,DC=X
changetype: add
cn: MQL*
objectClass: microfocus-MFDS-Resource
microfocus-MFDS-Resource-Class: MFESMAC
microfocus-MFDS-Resource-ACE: allow:SYSADM group:alter
microfocus-MFDS-Resource-ACE: deny:*:execute
microfocus-MFDS-UID: mfuid
#description: Allow full access any ESMAC MQ Listeners/Writers Screen

*****
```

3143258 (1112990)

- You can now use the `ECIResponse.getReturnCode()` method to obtain the return code for any errors from Enterprise Server.

3142092 (1113248)

- Communication with the console daemon has been improved. Messages are displayed more quickly and requests are being processed more efficiently and, as a result, times for initialization and shutdown might be reduced.

3136867 (1112483)

- Administrators can now add, delete or modify XA resources in the Enterprise Server Administration Web UI while a region is running.

2589624 (1085625)

Enterprise Server Auditing

[Back to the list](#)

- The `maxRetryTime` value in the audit configuration file now treats 0 as a no timeout time, and any negative number as an infinite timeout.

3150566 (1113592)

- Any extra information that was added to the syslog messages will now correctly appear in the structured data items.

Executables require relinking or recompiling

- Due to an internal change in version 4.0 of your product, you must at least relink any executable programs compiled prior to this version, to make them compatible with the latest run-time system. However, a full recompilation of your source code is the recommended action, to allow your executables to benefit from the product's latest programming and performance enhancements.

Relinking an executable without recompiling means using the original object code with the `cbllink` utility. Original object code is typically the binary file output (usually containing the `.obj` extension) produced during the original compilation process. An application can contain one or more binary files.

File Handling

[Back to the list](#)

- In some cases the ESF LDAP Security Administration Web Interface inadvertently removed users from groups when changing their password. This has been fixed.
3124294 (1111259)
- The ESF LDAP Security Administration Web Interface can now filter by Class and Resource name, description, and ACL. Previously, you could only filter on Class name.
2871549 (1106119)
- For .NET applications, set `XFHLOG=DEFAULT` to create the `XFHLOG` file in the current directory; otherwise it is created in `C:\ProgramData\Micro Focus\[ED/VS]\[Release]`. From release 4.0 onwards, this variable also has an effect for native applications.
3115116 (1110323)
- Setting the configuration option `ASCIISOSI=ON` will add the required SOSI characters to the relevant EBCDIC DBCS character strings, in order for them to be displayed or written out correctly.
3113802 (1110183)
- The OPEN mode of SYSOUT files now honors the DISP specified in the JCL.
3109432 (1109745)
- OPEN I-O of a virgin ESDS file now correctly returns a file status of 35, as it does on the mainframe.
2887724 (1108443)

MF Server Administrator (GUI)

[Back to the list](#)

- The total number of active sessions or clients in MFDS is now limited to 2000.

Micro Focus Directory Server

[Back to the list](#)

- The `mfd`s -g options D, O, and S have been added to the product Help.
2848627 (1102864)
- MFDS now disables and limits the scope of Web listeners on add. It also emits a warning if any insecure Web listeners are displayed in the validate and listener tables.

Run-Time System

[Back to the list](#)

- The run-time system now produces a more precise error message if a shared object of the wrong bitism is loaded.
- `scan64` is no longer available. This has been superseded by the COBOL Analysis functionality in the IDE.

Significant Changes in Visual COBOL 3.0

Visual COBOL version 3.0 includes significant changes in the following areas:

- [Compatibility AddPack](#)
- [Compiler](#)
- [Documentation](#)
- [Enterprise Server](#)
- [File Handling](#)
- [IDE](#)
- [Micro Focus Directory Server](#)
- [OpenESQL](#)
- [Reserved words](#)
- [SQL Option for DB2](#)

Compatibility AddPack for Visual COBOL

Compatibility AddPack for Visual COBOL is now deprecated and will not be available with release 3.0 and later.

The Dialog System GUI and run-time components and Dialog System Character Mode (on Windows and UNIX) which were part of the AddPack are now installed as part of Visual COBOL for Visual Studio. The run-time components are installed as part of COBOL Server. These are only included for backward compatibility and Micro Focus does not recommend that you use them for new development.

The other components which were part of the AddPack, the Character-Based Data File Editor, CSBIND and Screens, will be available upon request from Micro Focus SupportLine.

Compiler

- Replacing a partial token no longer causes the second part of the token to appear on a new line. This could happen if the new text was larger than the text being replaced.

2869185 (1105763)

Documentation

- There have been a number of new reserved words added to the language in this release; these are all in effect under MFLEVEL"19", which is the default level when running under the MF dialect. Any of the following words are now not allowed under default conditions, and you will need to remove/rename them, or specifically configure your environment to allow them: ALLOCATE FREE JSON END-JSON

Enterprise Server

- WEB CONVERSE now supports a value of 0 for the USERLEN and PASSWORDLEN options which matches the behavior on the mainframe. There is no change to the behavior of WEB SEND (client) which is to return LENGERR 139/140 when USERLEN or PASSWORDLEN are 0.

2989188 (1108602)

IDE

- Disabling CICS support from the properties of a file in your project within the IDE now correctly sets the NOCICSECM Compiler directive on that file.

File Handling

- A problem that generated a 39 error when attempting to access a VSAM file via an alternate index PATH element has been fixed.

2874622 (1106562)

Micro Focus Directory Server

- In the Enterprise Server Administration HTML GUI, the "Scripts" page functionality is only available if administration access is restricted and the logged on user has sufficient authority.

3101625 (1109025)

- Some additional CSRF security measures have been added to the Enterprise Server Administration HTML GUI.

3101205 (1108916)

OpenESQL

The new OpenESQL OPTIMIZECURSORS SQL compiler directive option is turned on by default for both ADO.NET (DBMAN=ADO) and ODBC (DBMAN=ODBC). For ADO.NET, this reduces cursor memory consumption thereby providing optimal performance. This also ensures that, for ODBC, embedded SQL cursors that use WITH HOLD and FOR UPDATE clauses have the same data integrity across all databases.

If your applications require the OpenESQL preprocessor to use the behavior provided in an earlier release, compile them using OPTIMIZECURSORS=NO.

Reserved words

- There have been a number of new reserved words added to the COBOL language; these are all in effect under MFLEVEL"19", which is the default level when running under the MF dialect. Any of the following words are now not allowed under default conditions, and you will need to remove/rename them, or specifically configure your environment to allow them:

ALLOCATE

FREE

JSON

END-JSON

SQL Option for DB2

- Help buttons previously available on the XDB Server Configuration Utility, XDB Service Controller, Options Dialog, Bind Utility, and Linker Config (Link Profile) UIs have been removed with the exception of error messages in the SQLWizard, Migrate, and Declaration Generator.

Significant Changes in Visual COBOL 2.3 Update 2

Visual COBOL version 2.3 update 2 includes significant changes in the following areas:

- [Enterprise Server](#)
- [Compiler](#)
- [MF Directory Server](#)
- [Monitoring and Management](#)
- [Run-Time System](#)

Compiler

- The ILPINVOKE directive is now allowed only in an initial \$SET. This is in line with similar directives like ILREF and ILTARGET. Previously, the ILPINVOKE directive was allowed in other positions in the source code, but could later result in 'Insufficient memory' errors.

2860347 (1104422)

- Replacing a partial token no longer causes the second part of the token to appear on a new line. This could happen if the new text was larger than the text being replaced.

2869185 (1105763)

- There is no longer a problem opening an RM/COBOL indexed file when the program has a RECORD CONTAINS n CHARACTERS clause and there are record descriptions with lengths less than n. This situation previously caused a 39 error on the OPEN (other than OPEN OUTPUT) because there was a mismatch in the minimum record length.

Enterprise Server

- Previously, it was possible to install groups that should not have been installed. If a group name, as defined in the Startup List, did not exist in the list of Groups then the next Group in the alphabetical order would be loaded instead. Now, if a Group is not defined in the list of Groups, a warning that the Group could not be loaded is issued.

2869848 (619107)

MF Directory Server

- The "-n" option for the mfd command now supports hostnames as the network addresses in addition to IPv4 addresses.

2816871 (1099564)

Monitoring and Management

- Messages that are written to the console log by applications that perform "display upon console" now contain a standard message ID (CASMG0001I).

2854207 (1103659)

Run-Time System

- The command_line_linkage tunable has been deprecated; equivalent functionality can be achieved by using the COMMAND-LINE-LINKAGE Compiler directive instead.

2838118 (1101539)

Significant Changes in Visual COBOL 2.3 Update 1

Visual COBOL version 2.3 update 1 includes significant changes in the following areas:

- [Data Tools](#)
- [Dialog System](#)
- [Editor Writing Assistance](#)
- [Run-Time System](#)
- [SQL: OpenESQL](#)
- [SQL Option for DB2](#)

Data Tools

- When filtering a data file, if there is no valid temporary directory set, you are prompted to set one using the option in the Preferences dialog box.
- The editor no longer allows you to open a file if the file size (without header size) is not a multiple of the record size on disk; an error is produced instead.
- The editor no longer allows you to open a file if the file size without header size is not a multiple of the record size on disk; an error is produced instead.
- The level numbers displayed in a record layout correspond to the levels used in the .idy file that was used when the structure file was created.

Dialog System

- Versions of the Micro Focus Compatibility AddPack released with version 2.3 of Enterprise Developer or Visual COBOL 2.3 or with earlier versions supported the use of the "MFOLECL_NO_THREAD_INIT" environment variable. Using this variable, you could disable the default OLE Class Library COM threading initialization. This helped avoid issues that could manifest as hangs and crashes, especially on Microsoft's Windows 8.x or 10 and with applications that are a hybrid between Dialog System, OLE class library and .NET elements - see http://community.microfocus.com/microfocus/cobol/visual_cobol/w/knowledge_base/20715.exception-occurs-when-native-dialog-system-program-calls-managed-winform.aspx.

In the version of the Micro Focus Compatibility AddPack released with Enterprise Developer 2.3.1 or Visual COBOL 2.3.1, this environment variable is now enabled by default and no longer needs to be set exclusively. To restore the previous behavior, use a new environment variable, MFOLECL_THREAD_INIT, and set it to Yes.

2848875 (1102920)

Editor Writing Assistance

- IntelliSense (Visual Studio) or Content Assist (Eclipse) suggestions are no longer offered if you start typing numbers and automatic triggering of suggestions is enabled.
- Pressing TAB in the Visual Studio editor now always inserts the highlighted item in the IntelliSense list of suggestions.

Run-Time System

- The Audit Manager contains a new TIMEOUT option. When a client sends an audit event using the 'CBL_AUDIT_EVENT' API, the event gets placed in the next available slot in a shared memory block. If shared memory is full (i.e. no slots are available), the event is re-tried until a slot becomes available.

If no Audit Manager is running, no events are removed from shared memory, and no slots will ever become available. Therefore, use the new TIMEOUT option so that a client will only retry sending until the TIMEOUT duration is reached; after which, it will stop sending audit events. If Audit Manager is recycled, events will start to be sent again.

To set the TIMEOUT for all Audit Manager clients, specify the following line in the Audit Manager configuration file:

```
mfaudit.timeout = n
```

Where n is the timeout value in milliseconds.

To set the TIMEOUT for an individual Audit Manager client, use the 'CBL_AUDIT_CONFIG_PROPERTY_SET' API. It takes an integer property-value, which should be the timeout value in milliseconds.

If TIMEOUT is set using both methods, the client property TIMEOUT takes precedence, unless this property is set to zero; in such cases, the TIMEOUT in the configuration file is used. If you use the 'CBL_AUDIT_CONFIG_PROPERTY_GET' API on the 'TIMEOUT' property, it only returns the TIMEOUT value for the client property; it does not return the value set in the configuration file.

2838689 (1101685)

- Several changes have been made to the implementation of IS DBCS, IS KANJI and IS JAPANESE class condition tests:

- IS [NOT] DBCS

When CHARSET"EBCDIC" is in effect, the IS DBCS test returns true when each character in the string is deemed to be a valid DBCS character. A valid character has its first byte in the range 0x41 through 0xFE, and the second byte in the range 0x41 through 0xFE, or the character is an EBCDIC space (0x4040). When CHARSET"ASCII" is in effect, the DBCS test uses an OS call to determine if the string contains only valid double-byte character, and returns true if valid.

- IS [NOT] KANJI

When CHARSET"EBCDIC" is in effect, the IS KANJI test returns true when each character in the string is deemed to be a valid Kanji character. A valid character has its first byte in the range 0x41 through 0x7F, and the second byte in the range 0x41 through 0xFE, or the character is an EBCDIC space (0x4040). When CHARSET"ASCII" is in effect, the IS KANJI test uses an OS call to determine if the string contains only valid Kanji character, and returns true if valid.

- IS [NOT] JAPANESE

When CHARSET"EBCDIC" is in effect, the IS JAPANESE test is not supported, and will generate a COBCH1806 Feature not supported in selected charset message on compilation.

When CHARSET"ASCII" is in effect, the IS JAPANESE test returns true when the string contains only double-byte Japanese characters or single-byte Japanese Katakana characters, and returns true if valid. When NSYMBOL"NATIONAL" is in effect, these class tests are not supported, and will generate a COBCH0303 Operand has wrong data-type message on compilation.

2812895 (1098401)

SQL: OpenESQL

- The DB2 CONCAT function and operator now convert to SQL Server using the HCOSS-supplied dbo.CONCAT for character, numeric and datetime data. If you are using BINARY or VARBINARY data, you must apply the HCOSS-supplied dbo.CONCAT_BINARY function. HCOSS applications deployed with earlier versions of Visual COBOL are affected, if they use string or binary concatenation. The mainframe dialect DB2 || operator and CONCAT function now call a new SQL Server scalar function dbo.CONCAT(). All existing programs with dialect=mainframe that use DB2 concatenation syntax should be recompiled. All existing SQL Server databases that are accessed by these programs must have dbo.CONCAT installed. To create the new function in your application's SQL Server database, you can either:

- Run a DSN bind against the customer database. Or:
- Execute the %ALLUSERSPROFILE%\Micro Focus\Enterprise Developer\hross\InstallDigitsFunction.sql script.

This is a one-time only change to the database.

2843818 (1102248)

SQL Option for DB2

- Spurious errors were sometimes returned while querying using an ALIAS.

2830383 (1100609)

Significant Changes in Visual COBOL 2.3

Visual COBOL version 2.3 includes significant changes in the following areas:

- [Building](#)

- [Converting Additional Directives to Projects' Properties](#)
- [CAS Security](#)
- [CAS XA Switch modules](#)
- [Compiler](#)
- [Data Tools](#)
- [File Handling - External File Handler](#)
- [IDE](#)
- [J2EE Connector](#)
- [MF Server Administrator \(GUI\)](#)
- [Updated Run-Time System](#)

Building

- Visual COBOL now supports Visual Studio parallel builds for COBOL projects. Parallel builds enable you to build multiple projects faster on multi-CPU machines.

If, after upgrading to this version of Visual COBOL, you start receiving unexpected build errors when compiling an existing multi-project solution, this may be a result of enabling support for parallel project builds. These are a couple of examples of issues that might be causing these errors:

- Using file references to project outputs in the same solution. You need to use project-to-project references instead.

Use **Project > Project Dependencies** to manage the project dependencies and build order within your solution.

- A customized build process such as one that is using pre- or post- build events.

If resolving any of these issues does not help resolve the build errors, consider disabling the parallel build support - click **Tools > Options > Projects and Solutions > Build and Run** and **set maximum number of parallel project builds** to 1.



Note: Parallel builds are not supported with Personal Edition licensing

Converting Additional Directives to projects' properties

Starting with this release, an **Update Project Properties** dialog box might start to appear when you are opening existing COBOL solutions. The dialog box recommends converting some of the directives specified in **Additional directives** to project's properties. This is to address an issue where you might try to set file properties that differ from the project directives and the directives specified in **Additional directives** that have a property equivalent take precedence over the file properties. As a result of this you might receive unexpected build issues when building your applications.

This dialog box helps eliminate the issue by converting the directives specified in **Additional directives** to project properties.

If you do not want the IDE to perform this check, click **Tools > Options > Micro Focus > General** and uncheck **Check Additional Directives for project properties**.

CAS Security

- The Enterprise Server External Security Facility now includes MLDAP ESM Module 2.0, with a new algorithm for identifying the best-matching resource-access rule and ACE for resource-access security checks. This algorithm is faster and matches most customers' expectations. The new algorithm also provides an optional "username substitution" feature. It can be enabled by setting "rule substitutions" to "yes" in the [Operation] section in the Security Manager configuration text area. When this is enabled, the string "\${user}" in a resource-rule name will be replaced with the name of the user that makes the request. For example, a DATASET rule named "USERS.\${user}.*" would apply to datasets with the requesting user's name as the second qualifier. In rare cases, customers with complex, ambiguous resource-access security rules might see experience changes in behavior as a result of the new

algorithm. The old algorithm is still supported and can be enabled by setting "version 1 authentication" to "yes" in the [Operation] section of the Security Manager configuration.

2807531 (1097783)

CAS XA Switch modules

- The XA switch modules now support dynamic registration.

2682101 (1092325)

- The XA switch modules now support batch-only operations when multiple XA Resource Managers have been defined.

2664675 (1091082)

- In Visual COBOL 2.2 update 2, Micro Focus identified undefined run-time behavior when the following combination of directives was specified: SIGN"EBCDIC", CHARSET"ASCII", and one of the following: HOST-NUMMOVE, HOST-NUMCOMPARE or SIGN-FIXUP. Previously (Visual COBOL 2.2 update 1 and earlier), if this combination was specified, the SIGN"EBCDIC" directive should have been ignored, to avoid a mixture of ASCII and EBCDIC characters; however, SIGN"EBCDIC" was still being honored, resulting in undefined run-time behavior. Therefore, this combination of directives is now invalid for Visual COBOL 2.2 update 2 or later, and if specified, will be rejected at compile time.

2786397 (1095265)

Compiler

- For native COBOL, the size limit of the Data Division now stands at 2GB -1.

2796076 (1096384)

- COBDATA has no effect on compilation. The output of the Compiler is the same location regardless of whether COBDATA is set.

Previously, it was not possible to specify sign(EBCDIC) with sign-fixup, host-num-move or with host-num-compare. This combination is now supported in native COBOL but remains invalid for managed COBOL code. This is applicable to version 2.2 U2 HotFix 10 onwards.

2824577 (1100823)

Data Tools

- DFCONV now returns the correct return-code; previously, it would always return 0.

File Handling - External File Handler

- Custom file handlers (using DYNREDIR) are now called for each part of a concatenated file.

2795077 (1096322)

IDE

- Changes in the C compiler in Visual Studio 2015 affect the way you link COBOL object code and C object code built with that version of Visual Studio in the same executable. In this scenario, you must use the Microsoft link utility and the C runtime libraries directly from Visual Studio, rather than the Micro Focus cblink utility, the Microsoft link utility and the libraries supplied with Visual COBOL. You might also need to specify some additional C runtime libraries - see the Microsoft documentation for more details.

Note that when using COBOL and C object code together, Micro Focus recommends you build and maintain the COBOL and C executables as separate projects, and use import libraries and the Micro Focus C functions for calling COBOL (see "C functions for calling COBOL" in the product help) to resolve calls between them.

- The default warning level for new COBOL projects in Visual COBOL for Visual Studio is now "Include recoverable errors (Level E)". This also includes "Severe errors only (Level S)" and unrecoverable (Level U) errors. Micro Focus recommends you set the warning level on the COBOL page in a project's properties to "Include warnings (Level W)" where appropriate to help avoid potential coding problems.

J2EE Connector

- Visual COBOL version 2.3 provided a new command-line argument to Java, mf.ssl.algorithm, which can be set to an appropriate algorithm.

2799213 (1096684)

MF Server Administrator (GUI)

- Passwords that entered through either the MFDS or the ESMAC interface now use the same encoding.

2792382 (1096011)

Updated Run-Time System

- COBOL Server now provides an execution environment capable of running applications that were each built using different development products. A consequence of this is that if your application has a main COBOL executable (.exe) that was built with a version of Visual COBOL prior to version 2.3, you should ensure that the executable is rebuilt and packaged with the new run-time system. You can rebuild from the IDE or the command line.

Other COBOL subprograms built with previous versions of Visual COBOL are not required to be rebuilt.

Significant Changes in Visual COBOL 2.2 Update 2

Visual COBOL version 2.2 update 2 includes significant changes in the following areas:

- [Compiler](#)
- [Compiler Front-end](#)
- [Documentation](#)
- [J2EE Connector](#)

Compiler

- When using the HOSTRW directive with the mainframe dialect, Report Writer will now produce the full range of ASA control characters and will emulate mainframe print files.

2697615 (1094527)

Compiler Front-end

- Fixed Binary (p<=7) is now an 8-bit, signed, 2's complement binary integer by default.

Documentation

- The default setting for the MFALLOC_PCFILE environment variable has changed; the default is now set to Y, which means that when cataloguing a file that has a DCB attribute of DSORG=PS, a physical file is created for it if one does not exist. Previously, the default was set to N, which meant that a file was not created.

2697571 (1094370)

J2EE Connector

- The listSystem.properties file in package com.ibm.ctg.client was missing documentation for some sections.

(606556)

Significant Changes in Visual COBOL 2.2 Update 1

Visual COBOL version 2.2 update 1 includes significant changes in the following areas:

- [.NET Compiler](#)
- [SQL: COBSQL](#)

.NET Compiler

- In member reference in managed COBOL syntax, you may now only use parentheses when referencing methods. You can no longer specify parentheses when referencing fields or properties, as this will produce a syntax error. For example:

```
set intLength to testString::Length()
```

must change to:

```
set intLength to testString::Length
```

SQL: COBSQL

- COBSQL now displays appropriate COBOL syntax errors after encountering EXEC SQL statement errors.

2673619 (1093197)

Significant changes in Visual COBOL 2.2

Visual COBOL version 2.2 included significant changes in the following areas:

- [.NET Compiler](#)
- [CCI Session Layer Code](#)
- [Compiler](#)
- [Interface Mapping Toolkit](#)
- [MF Directory Server](#)
- [MF Server Administrator \(GUI\)](#)
- [MFBSI](#)
-
-
-
-
-
-
- [Request Handler](#)
- [Samples Browser](#)
- [Visual Studio IDE](#)

.NET Compiler

- The use of ILSMARTLINKAGE in sub-programs is now working correctly. In previous versions, if it was specified in a sub-program, a dynamic call of that program from another program could cause execution failure.

2608825 (1087435)

- In managed code, the statement DISPLAY UPON SYSERR now behaves as expected. Previously, it was behaving the same as DISPLAY UPON CONSOLE.

2589053 (1085576)

- When the PROPERTY keyword is used on a numeric edited field, the SET method now works correctly when the program specifies DECIMAL POINT IS COMMA.

2588508 (1085560)

CCI Session Layer Code

- A new option, use_global_namespace, is available for the cci.ini file in the Windows %SystemRoot% folder. If use_global_namespace is set, all the ccishared memory objects are created in a system-wide address space, and the applications hosted by different users, including system services, can communicate. To use this facility, edit the cci.ini file and ensure use_global_namespace is set to "yes". [ccismem-base] # Allow interaction between users on a # single system. Using this option reduces security as # all users will have access to the same name space. use_global_namespace=yes If the value of this option is anything other than "yes", or if the option is missing, no change is made to the existing behavior.

2195519 (1062800)

Compiler

- The default for the NSYMBOL directive under DIALECT(ENTCOBOL) has been changed to NSYMBOL(NATIONAL) to emulate the equivalent IBM default.

2657471 (1090355)

- To improve RM/COBOL and ACUCOBOL compatibility, the SIGN clause at a group level is no longer applied to non-DISPLAY usage signed numeric data items within the group, just as it is not applied to unsigned numeric data items and non-numeric data items within the group.

2549904 (1082171)

- Previously, even though no code was generated, the Compiler allowed the ON EXCEPTION and NOT ON EXCEPTION phrases in the DISPLAY statement in formats that do not allow these phrases. As a result, if the DISPLAY statement was in the ON EXCEPTION phrase of another statement, the NOT ON EXCEPTION phrase would bind incorrectly with the DISPLAY statement instead of with the intended containing statement - for example, ACCEPT or CALL.

Interface Mapping Toolkit

- For program-based Service Interfaces, if the program-id name in the COBOL source is in lowercase and is not surrounded by quotes, its corresponding entry-point name is now forced to uppercase when used in a Service Interface Operation. Previously, the case was preserved. As a result of this change, existing Service Interfaces will become invalidated if you refresh their program's annotations because of the new spelling of the entry-point name. To avoid this, you need to surround the program-id name in the COBOL source with quotes before you refresh the annotations.

MF Directory Server

- The mfd command line option for exporting registered Enterprise Server definitions to an XML file now supports the "*" option. This exports all registered servers rather than a specified server. Multiple server definitions are now exported into the target directory and saved into a file with the default name ALLSERVERS.xml. The import option now also supports the import of multiple server definitions from a single XML file.

2641890 (1088838)

- mdump now supports a new option, -e, to help you query the Security Manager configuration details. The possible values of the option are: "1" - shows security configuration that applies to any returned

enterprise servers; "2" - shows security configuration for MFDS and the default Enterprise Server security configuration. This requires MFDS version 1.15.00 or higher; "3" - returns the properties of all configured external Security Managers.

2487164 (1081693)

MF Server Administrator (GUI)

- When adding a user to an external security manager, you can now include a password expiry time in the Advanced Configuration section of the Add New User wizard in Enterprise Server Administration. The field value is specified using generalized time format (YYYYMMDDHHMMSS.0Z), and can be used by the MLDAP ESM for calculating whether a user's password has expired and requires updating. This value may only be specified using this page when adding a user. You need to use an external directory services configuration tool to edit it.

2562118 (1083203)

Request Handler

- A problem that caused BIS to create log files in a directory named C:\ProgramData\AcuCorp\BIS\LogFiles was fixed. BIS no longer creates log files unless specified and the BIS logging service is now disabled by default. To enable it, you need to use the following global environment variable:

BIS_LOG=[OFF | ON | <directory>] Where the values are:

- OFF - disables logging (the same as if BIS_LOG is not specified or is left blank)
- ON - enables logging and directs the log files into the default location, which must not be read-only.
- <directory> - enables logging and directs the log files into the specified directory. The user must ensure that the BIS request handler has write rights for this directory. The directory must be an absolute path or network path. If the specified directory does not exist, BIS will attempt to create it. The containing directory must exist.

The BIS_LOG variable is only examined when the BIS application pool is started or recycled. After setting or changing BIS_LOG, IIS must be restarted in order for the variable to take effect.

Samples Browser

- If you have installed the version of this product for both Visual Studio 2010 and Visual Studio 2012, the Samples Browser now provides links for you to open the samples in either one of them.

Visual Studio IDE

- Performance when loading and building COBOL projects that consist of a large number of files has been improved.

2657121 (1090316)

- When building a project outside of the IDE, adding a new file to a project no longer results in a full rebuild of the project.

2617003 (1089623)

- You can now change the display format of values on individual rows in the Watch window using modifiers after the names of variables and expressions. To do this, click a row, press F2, and type a modifier after the name of the variable or the expression as follows - type VariableName,h or VariableName,x to always display the variable or the expression in hexadecimal format; type VariableName,d to always display numeric variables in decimal format and strings - as text.

2614182 (1087959)

- When you perform a search in the IDE for copybooks that your programs depend upon, the COPYEXT directive you set from the Additional directives field on the COBOL tab in your project's properties is now used in preference to the list of copybook extensions specified in Tools > Options > Text Editor >

Micro Focus COBOL > Advanced, and in Copybook extensions. This change does not affect the project build and background parsing that already use COPYEXT.

2612053 (1087773)

- Attempting to restart an enterprise server failed if the process took longer than fifteen seconds.

2607051 (1087259)

- When you debug native code and query a data item which contains null bytes, the value displayed in the Watch window is no longer truncated at the first null byte.

2604749 (1087235)

- A new setting, "Define DEBUG constant", is now available on the COBOL page in the project properties of managed projects for the Debug configuration. You can select it to add "constant"DEBUG(1)" to the build directives.

2600567 (1086629)

- The Output window now displays a notification when the compilation of the IMS files has been successful.

2600137 (1086667)

- A problem with the value of "Link with objs" setting being duplicated after you reload the COBOL Link properties page has been resolved.

2595408 (1086091)

- IntelliSense now displays data names longer than 32 characters correctly.

2594901 (1086053)

- Previously, when you upgraded COBOL projects with signed assemblies from Visual Studio 2003 format to Visual Studio 2010 format, the signed assembly property was lost.

2585458 (1085258)

- There is an improvement in the performance of the cursor in the text editor when working with larger files and projects.

2585450 (1085255)

- Using linked files in projects has been improved as follows:
 - Folders that only contain linked files no longer disappear from the project when you reload it.
 - The location of linked files is preserved in the project structure when you reload the project.
 - Moving a linked file within a project no longer moves the actual file on the disk but only moves the link inside the project structure.
 - It is no longer possible to copy a linked file into the same project where, previously, this created a copy of the actual file on the disk.

2574102 (1084208)

- When you convert Net Express projects that compile to .int or .gnt files to Visual COBOL for Visual Studio, the application environment variables are now imported successfully.

2569777 (1083953)

- It is now possible to convert Net Express projects that do not contain any COBOL source files to Visual Studio projects.

2568638 (1083900)

- You can now set environment variables used by the native COBOL Run-Time system during initialization from the project properties - click Environment on the Application tab.

2539447 (1081251)

- Pressing Enter to create a new line now preserves the indent in the COBOL editor in Smart edit mode.

2496411 (1077769)

- Pressing End inside the writing area of the COBOL editor now positions the cursor on the first non-blank character, if columns 12-72 contain any text. If the line is blank, the cursor is positioned on column 73.

2496351 (1077767)

- The COBOL editor now supports a "Smart edit mode" (see Tools > Options > Text Editor > Micro Focus COBOL > Margins) which is enabled by default. In Smart edit mode, when you press ENTER in the code (A/B) area the comments move to the next line and stay in the right-hand area.

2496350 (1077768)

- You now receive a suitable error message when you try to use "Start Without Debugging" with Mainframe Subsystem Application projects.

2495298 (1078343)

- You can now use Solution folders to group projects in your solution.

2259310 (1065760)

- If, when you start debugging, the debug target does not exist you now receive an error message that correctly states the missing filename.
- Any leading or trailing spaces you added to the name specified in the "Output name" field on the Application tab in the properties of a native project are now ignored.
- Class view no longer contains outdated information from previous background syntax checking after classes have been renamed.
- Class view no longer contains outdated information from previous background syntax checking when the files defining types have been removed.
- A problem with navigating to member definitions from the Class View has been resolved.
- A problem with navigating to member definitions from the Class View has been resolved.
- An issue where canceling some of the dialogs in Tools > Options resulted in the changes being saved anyway, or the pages not reverting when next loaded, has been resolved.
- You can now start and stop servers from the Server Explorer when using the local machine's IP address or DNS name.
- It is now possible to debug a project that has its "Compile for debugging" setting enabled in the project properties for the Release configuration.
- Selecting a project in Class View now triggers background syntax checking, if it has not been performed for that project yet.
- If you have .NET Framework v. 4.5 installed, creating a COBOL Web application with Visual Studio 2010 Shell was failing with a Signal 114 error and was causing the IDE to crash. This issue has been resolved for the IDE. To work around the issue when you compile COBOL Web Applications from the command line, you need to manually set the environment variable VisualStudioVersion to 10.0 before compiling.
- Using the tab character in comments no longer causes issues with the colorization of the code.
- The Visual Studio IDE no longer crashes if you try to close it while directives scan is in progress.
- Adding files to a project when directives scan is disabled was taking a long time to complete.
- Friend assemblies (as identified by the InternalsVisibleToAttribute) are now correctly handled during syntax checking, and types are correctly found within friend assemblies when they are present.
- An issue with adding new platforms or configurations to a project or solution when the project contains pre- or post-build events has been resolved.
- A problem with navigating to member definitions from the Class View, when the definitions occurred within copybooks used within partial classes, has been resolved.
- A problem where trying to navigate to the source of an item from the Class View positioned you on the wrong line, has been resolved.
- An issue where the Implements smart tag was present for interfaces that were fully implemented (when some of the methods from that interface were implemented explicitly; and in addition to another interface which was not fully implemented) has been fixed.
- An issue where method signatures were rendered incorrectly if the member had a 78 level item in its local storage has been resolved.

Significant Changes in Visual COBOL 2.1 Update 1

Visual COBOL version 2.1 update 1 includes significant changes in the following areas:

- [Documentation](#)

Documentation

- To ensure no loss of functionality when accessing Vision and RM/COBOL data files, you should use the appropriate IDXFORMAT Compiler directive setting or file handling option, and not use the CALLFH(ACUFH) Compiler directive. See 'Configuring Access to Vision Files' and 'Configuring Access to RM/COBOL Data Files' for more information.

Significant Changes in Visual COBOL 2.1

Visual COBOL version 2.1 includes significant changes in the following areas:

- [Compiler](#)
- [IDE](#)
- [Run-Time System](#)
- [Vision File System](#)

Compiler

- The use of extended ACCEPT or DISPLAY statements in Windows applications (ilsubsystem"2") now produces error COBCH1634: Extended ACCEPT/DISPLAY not allowed with a Windows GUI executable (ILSUBSYSTEM"2"). Remove the syntax or change the output type of your application to 'Console Application'.

2575702 (1084365)

IDE

- A problem with the item templates in the Add New Items dialog box not available for PL/I projects has been resolved.

2581187 (1084876)

- There is now a build summary report for projects that compile to multiple output files.

2578006 (1084506)

- Previously, Visual Studio used to always rebuild the entire project after you changed the project properties or added files when there were build error. This has been changed so that during subsequent builds Visual Studio only rebuilds the files that are out-of-date.

2576400 (1084393)

- A new setting has been added to the COBOL tab in the project properties, "Generate listing option", which produces an .lst file in a Listing subfolder in the project directory. The "Generate directives file" setting now generates the directive file in the project output directory.

2494484 (1077662)

- Command line arguments are now passed to the program when started via the 'Start without Debugging' option in the Visual Studio IDE.
- Previously, if you started debugging using CBL_DEUGBREAK and there was a COBOL source file open in the editor, when you stopped debugging its dictionary file (.idy) was not being released. This resulted in any subsequent builds and rebuilds failing. This problem has been resolved.
- The native COBOL projects in Visual Studio have been enhanced so that you can now build them to multiple executables - each COBOL program in the project producing a separate executable.

- You no longer receive an error message when adding an event handler to an object in a COBOL WPF project if the .xaml code behind file includes a class that does not have methods in it.
- The IDE no longer ignores certain Run-Time System errors from the Compiler which previously caused the build to fail without an explanation.
- Compile extensionless COBOL files with the IDE no longer fails.
- Using the UI to add COBOL switches or environment variables to a COBOL Web Application (right-click the configuration file in Solution Explorer and select Edit) no longer causes a configuration error when you run the application.

2558133 (1082871)

Run-Time System

- When running a full-screen application inside a terminal emulator on Linux, the actual size of the terminal is read at startup and reread when the terminal is resized. This behaviour is also supported on AIX, HP/UX, and Solaris. The Micro Focus vt220 terminfo entry now correctly describes a 24-line display. A vt220-25 terminfo entry is included for compatibility with the previous behaviour.

2579335 (1084817)

Vision File System

- When you configure your application to return RM/COBOL file status codes, by setting COBFSTATCONV=rmstat, the codes returned are ANSI'85 codes.

2553438 (1082469)

Significant Changes in Visual COBOL 2.0

Visual COBOL version 2.0 includes significant changes in the following areas:

- [Compiler](#)
- [DB2](#)
- [File Handling](#)
- [IDE](#)
- [OpenESQL](#)
- [Run-Time System](#)

Compiler

- An attempt to serialize a class containing COBOL data such as PIC X or group items could lead to a serialization exception.

2555177 (1082730)

- Dependent assemblies (ones that are required by other assemblies) that fail to load now issue a COBCH1581 warning "Could not load dependent assembly AssemblyName". Previously, they generated a COBCH0942 error. If an assembly generates both COBCH0942 and COBCH1581 errors, the error report lists only the COBCH0942 error.

2507380 (1078845)

- The scope of the ILUSING Compiler directive when used in a \$set command has changed. The scope of the directive is now limited only to the source file it is set in, and not globally. This new behavior may mean that your source files no longer compile. To resolve this, add the required ILUSING statements to the required individual source files, or add the ILUSING directive on the command line. Alternatively, use the IDE to achieve the required behavior: in Visual Studio, use the Namespaces tab; in Eclipse, set the directive in the Additional Directives field.

DB2

- The DB2 ECM has been updated to resolve run-time errors returned when compiling against mainframe databases in 64-bit mode.

2549058 (1082441)

File Handling

- When reading a file cataloged as DISP=SHR the file handler now buffers the read for better performance.

2518330 (1079491)

IDE

- Using File > New > Solution From Net Express Project to convert a Net Express project to a Visual COBOL solution on a machine with a language locale other than English or Japanese produced an empty solution.

2548551 (1082073)

- The COBOL tab in the project properties of a new WPF Application sometimes was not showing any settings.

2535705 (1080950)

- In some menus and dialogs in the IDE copybooks were referred to as copy files. They are now consistently referred to as copybooks.
- The project properties have been enhanced. The SQL directives are no longer shown in Build Settings on the COBOL page where they sometimes appeared as out-of-date. The SQL directives now appear only on the SQL page.
- Adding a watch on an item of the form `buffer(1:size)` caused the IDE to hang when "size" was not initialized.
- ActiveX components in the Windows Forms of a managed project could cause the programs to crash at run-time. Previously, to work around this problem, you had to check the ActiveX references in the project and set their CopyLocal reference property to True. This is now done automatically when the references are added.

2536071 (1080978)

- Adding an existing Windows Form to another project as a linked file did not function correctly. When you reopened the project, the designer and the resx files were missing from all Windows Forms in the project.

2524313 (1079984)

OpenESQL

- OpenESQL did not always return a consistent error message or it returned an incorrect error message when a data conversion exception occurred within the ADO.NET run-time system.

Run-Time System

- On Windows 7, building 64-bit native COBOL applications always rebuilt the entire project. This was caused by the Microsoft FileTracker tool, used by the COBOL projects to track dependencies, not working because of Windows 7 security updates. This product now includes a fix to work around this issue.

Unsupported or Deprecated Functionality

The following topics describe functionality that was removed or deprecated at each product release.

Unsupported or Deprecated at Visual COBOL 4.0

There are no features or functionality unsupported or deprecated at version 4.0.

Unsupported or Deprecated at Visual COBOL 3.0

The following features or functionality are no longer supported or are deprecated at version 3.0:

- The HOSTSIGNS Compiler directive is no longer supported. Micro Focus recommends that you use the following Compiler directives instead: SIGN-FIXUP, HOST-NUMMOVE, and HOST-NUMCOMPARE.
- Compatibility AddPack for Visual COBOL - this is now deprecated and will not be available with release 3.0 and later.

The Dialog System GUI and run-time components and Dialog System Character Mode (on Windows and UNIX) which were part of the AddPack are now installed as part of Visual COBOL for Visual Studio. The run-time components are installed as part of COBOL Server. These are only included for backward compatibility and Micro Focus does not recommend that you use them for new development.

The other components which were part of the AddPack, the Character-Based Data File Editor, CSBIND and Screens, will be available upon request from Micro Focus SupportLine.

- Audit Manager is deprecated and provided for backward compatibility only. We recommend that you use syslog events instead. See *Enterprise Server Auditing* for more information.
- The following DB2 environment variables are deprecated at version 3.0, and provided for backward compatibility only.
 - HCOBND - Micro Focus recommend you use either the BIND or the BINDDIR compiler directive option.
- The following compiler directives are deprecated at version 3.0, and provided for backward compatibility only.
 - CONVERTRET
 - IDYSRCPATH
 - ILOBJECTIFY
 - OPTION
 - SPZERO
 - TRICKLE
- The following file handling options are deprecated at version 3.0, and provided for backward compatibility only. Micro Focus recommend you use IDXFORMAT"8" instead:
 - STRIPING
 - MAXSTRIPEDIGITS
 - MAXSTRIPEFILES
 - MAXSTRIPESIZE
 - STRIPE-X
 - STRIPENAMETYPE -

Unsupported or Deprecated at Visual COBOL 2.3 Update 2

The following features or functionality are no longer supported or are deprecated at version 2.3 Update 2:

- The command_line_linkage tunable has been deprecated; equivalent functionality can be achieved by using the COMMAND-LINE-LINKAGE Compiler directive instead.

Unsupported or Deprecated at Visual COBOL 2.3 Update 1

The following features or functionality are no longer supported or are deprecated at version 2.3 update 1:

Windows Azure versions

[Back to Top](#)

- Starting with Visual COBOL 2.3 Update 1, versions of the Microsoft Azure SDK earlier than version 2.8 are no longer supported.
- Support for the Microsoft Azure SDK is no longer provided in Visual COBOL for Visual Studio 2012. If you have COBOL Azure projects created with earlier versions of Visual COBOL for Visual Studio 2012, to maintain them, use Visual COBOL for Visual Studio 2013 or 2015.

Unsupported or Deprecated at Visual COBOL 2010

The following Net Express features or functionality were not supported or were deprecated from the first version of Visual COBOL:

These Net Express compiler directives are not supported by Visual COBOL :

01SHUFFLE
64KPARA
64KSECT
AUXOPT
CHIP
COBIDY
DATALIT
EANIM
EDITOR



Note: The directive SPZERO was already deprecated from Net Express 5.0 onwards, and is provided for backward compatibility only. You should instead use *SIGN-FIXUP*.

and the pseudovariabes of the following Net Express environment variables are obsolete and cannot be used:

BASENAME
FILENAME
PATH
TARGETDIR

Upgrading from Net Express to Visual COBOL

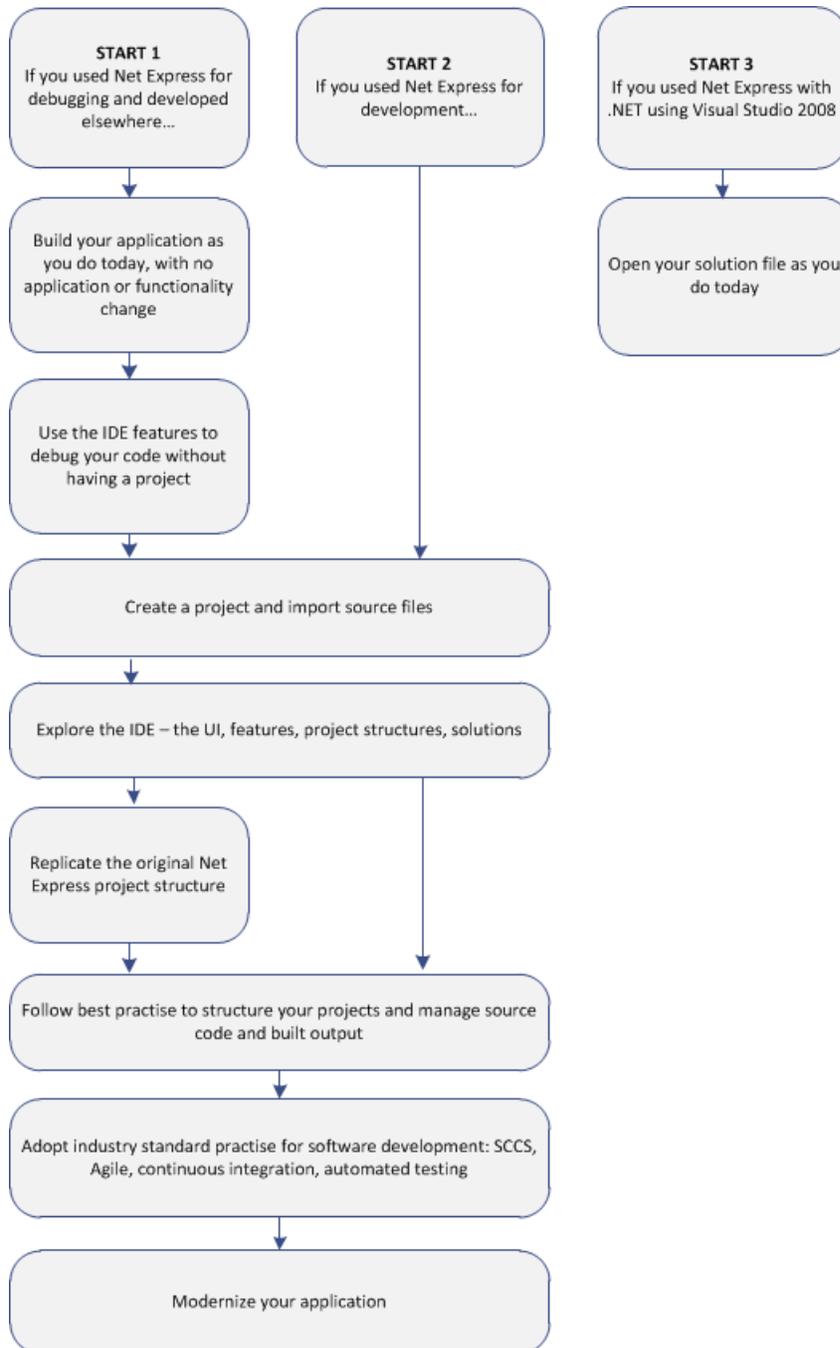
The following topics show you the process of moving existing Net Express applications into Visual COBOL for Visual Studio.

An introduction to the process of upgrading your COBOL applications

The following topics show you the process of moving existing Net Express applications into Visual COBOL for Visual Studio. This information assumes one of the following starting points:

- You currently use Net Express just for debugging, and edit files and compile projects by other means (START 1)
- You currently use Net Express for all your development tasks (START 2)
- You currently use Net Express for .NET in Visual Studio 2008 (START 3)

The steps to move to Visual COBOL are illustrated below:



After following this process you will be able to use the Visual COBOL for Visual Studio features to improve development and modernize your applications.

Compile at the Command Line Using Existing Build Scripts

Application executables that were compiled using earlier Micro Focus products must be recompiled from the sources using Visual COBOL. If you do not recompile, you may receive an error. The exact error depends on the operating system you are running.

Most Net Express projects should compile cleanly using your existing build scripts and makefiles without any changes to your code, as Visual COBOL can use the `cobol` and `cbllink` commands to create `.int` and `.gnt` files. By specifying the `ILGEN` compiler directive you can also use these commands to create `.NET-compatible .exe` files, or use the `JVMGEN` directive to create `JVM-compatible .exe` files.

Fixing compilation issues

You might encounter some problems when compiling your Net Express applications in Visual COBOL.

Micro Focus continues to enhance the COBOL language, for example, by expanding the list of reserved COBOL words and adding new keywords to it as part of new levels of the COBOL language (each Micro Focus release corresponds to a particular level). Applications created with an older Micro Focus product might use data names that are now reserved keywords in Visual COBOL, which can result in a COBOL syntax error COBCH0666 ("Reserved word used as data name or unknown data description qualifier"). See [Reserved Words Table](#) for a comprehensive list of reserved words and level at which they are supported.

Also, these Net Express compiler directives are no longer supported:

```
01SHUFFLE
64KPARA
64KSECT
AUXOPT
CHIP
COBIDY
DATALIT
EANIM
EDITOR
```

and the pseudovariabes of the following Net Express environment variables are obsolete and can't be used.

```
BASENAME
FILENAME
PATH
TARGETDIR
```

You should consider using the following methods to solve these problems:

- Rewrite the source to avoid using these keywords in your code and directives files.
- Use the `REMOVE` Compiler directive to remove individual keywords from the reserved words list.
- Use the `MF` or `MFLEVEL` compiler directive to select an earlier version of Micro Focus COBOL that your code is compatible with. For example, setting `MFLEVEL"12"` ensures compatibility with Mainframe Express 3.0 and 3.1; Net Express 4.0, 5.0, and 5.1; and Server Express 4.0, 5.0, and 5.1. Refer to [Reserved Words Table](#) for the value to use to ensure support for your existing reserved words.

Setting `REMOVE` and `MFLEVEL` directives from the command line

To use `REMOVE` from a Visual COBOL command prompt, type the following:

```
cobol myprogram.cbl remove(title) ;
```

The command above removes `TITLE` as a keyword from the language so you can use it as an identifier in a COBOL program.

To use the set of reserved words that was used for Net Express v5.1 WrapPack 5, use this command line:

```
cobol myprogram.cbl mflevel"15" ;
```

Setting `REMOVE` and `MFLEVEL` directives in the source code

To set either one of the directives in your source code, type the following starting with `$` in the indication area of your COBOL program:

```
$set remove "ReservedWord"
```

Or:

```
$set mflevel"nn"
```

Single-threaded run-time system

The single-threaded run-time system is not available in Visual COBOL on Windows. Instead, both single-threaded and multi-threaded applications run using the multi-threaded run-time system. This has no effect on your existing applications.

Debugging Without a Project

Having compiled your existing code into the required format, it is possible to debug your code using the debugger in the same way that you did with Net Express, even before you create a Visual COBOL project in the IDE and import the code into it (although with the lack of a project, elements of the program have no context and the scope of debugging is limited).

You can cause debugging to be triggered at a specific point in your code by using the `CBL_DEBUGBREAK` and `CBL_DEBUG_START` library routines. You can also use the `debug_on_error` runtime tunable to enable the debugger to start when your a running program terminates with a run-time system error.

Run your program. When the routines or tunable trigger debugging, Visual Studio starts, displaying the source file at the current line of code being executed. You can then make use of the debugging features of Visual COBOL which include:

- Step into the next statement at the current line of code and suspend execution.
- Step over the next statement at the currently executing line of code without entering it, and suspend execution. The method will be executed normally.
- Return from a method or paragraph that has been stepped into, and suspend execution. The remainder of the code inside the method is executed normally.
- Resume execution of the program from a suspended line of code.
- Display values of all variables contained on the current execution line.

Create a project and import source

Follow these steps to use your source files in a new project in Visual COBOL.

1. Start Visual Studio.
2. Click **File > New > Project**.

You'll see a list of **Installed Templates** on the left. Expand **COBOL** and choose **Native**.

This gives you a list of project types. The main difference between these types is the nature of the artefacts they build, and after creating a project, you can easily change its type and output accordingly.

- **Windows Application** - creates a project that builds a single executable `.exe` by default, and is best used for graphical applications.

- **Console Application** - creates a project that builds a single executable `.exe`, and is best used for character-based applications that use the console subsystem. You can configure it to build an `.exe` file for each source program.
- **Link Library** - creates a project that builds a single `.dll` file.
- **INT/GNT** - creates a project that, by default, outputs one `.int` file for each of your source programs. You can change the build order to `.gnt` by right-clicking the project in Solution Explorer and choose **Properties**, select the COBOL tab, and choose **Compile to .gnt**.

The other fields in this dialog box specify the folder structure in which your project will be placed:

- **Name** - the name of the project.
- **Location** - the folder in which the project will be created. If you specify a folder that doesn't exist, Visual Studio will create it.
- **Solution** - a solution is a container in which you can group logically-related projects. Only one solution can be open in Visual Studio at a time. At this stage you can either create a new solution that will use the name specified, or add the project to the solution currently open in Visual Studio.

You can select **Create directory for solution** in order to give the solution a different name to the project name. This is useful when you are likely to have several projects in the same solution.

3. Right-click your project in Solution Explorer and select **Add > Existing Item**.
4. Click **Add** and navigate to the folder containing the files you want to add to the project.
5. Choose the files you want to add and then click **Add**.

Those files are then added to the project in Solution Explorer. These files are copied, not moved, to the project folder in the file system. If you click the down arrow on the **Add** button, you can choose **Add as Link**, which adds a reference to the file in the project but neither moves or copies the original. Added files have the icon ; linked files are indicated by the icon .



Note: If you right-click your project in Solution Explorer and choose **Add Existing COBOL Items**, you choose a folder instead of individual files. All files in that folder with the extensions listed in the Specify Source Files page of the import wizard are then added to the project in Solution Explorer. You can only add files as links using this method.

Adding copybooks

You can add copybooks to your projects in the same way as COBOL files, by right-clicking your program, choosing **Add > Existing Item** and browsing to a copybook. However, it is not compulsory to add copybooks to your project. You can set the copybook dependency paths for your project from the **Project Properties > Copybook Paths** page. Copybooks are not compiled at build time due to the file's **Build Action** property being automatically set to **None**. (You can also set this property for COBOL source files too, to keep a file in the project but not include a built version in any output.)

By default, Visual COBOL identifies files as copybooks by their `.cpy` extension. You can specify other file extensions as copybooks in the IDE preferences - click **Tools > Options > Text Editor > Micro Focus COBOL > Advanced > Copybook Extensions**, and enter the additional values in the text box. Alternatively, you can add the copybook with unknown extension to your project and then reference the file from within a COBOL program using the COPY statement. Visual COBOL then recognizes that extension as a copybook but only across the current solution.

Setting Compiler directives

Some Compiler directives are set on project creation, and differ between the Debug and the Release configurations. To add directive to your project, right-click on the project in Solution Explorer and choose **Properties**. On the **COBOL** tab, you can see directives that are set by the IDE in the **Build Settings** text box. Enter others in the **Additional Directives** text box as a space-separated list.

If you use a separate text file to manage your directives, you can reference this instead by entering the `USE"directives file"` directive. You should enter a path relative to the project directory.

Building the project

Having added all the files and made any necessary configuration changes, you can compile and link the COBOL source and generate the output. Right-click the project in Solution Explorer and click **Build**.

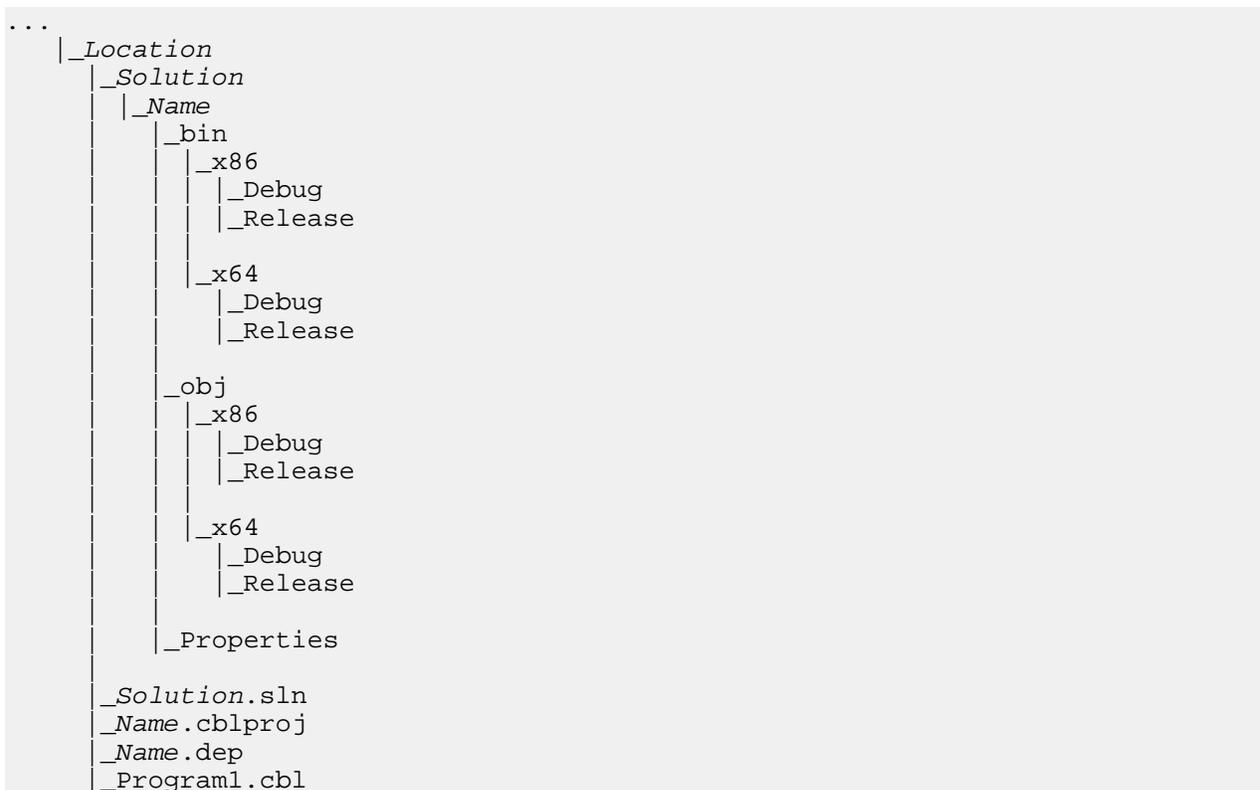
If your source code contains tab stops compilation might fail, as while a COBOL tab is eight characters long, the IDE's tab is four characters long, and lines of code might be starting in the sequence number and indicator areas section (columns one to seven) of the program instead of from column eight.

You can fix this problem using the `SOURCETABSTOP(n)` compiler directive, where *n* is the number of space characters by which to expand tab characters during compilation.

Using Visual COBOL for Visual Studio

Understanding the structure of Visual COBOL solutions

On creating a new project, the following files are created in the file system with the following structure:



If you select the Create Directory for Solution option when creating a solution, the structure is slightly different.

In the *Solution* folder:

- **Solution.sln** - a description of the solution and what it contains.
- **Name.cblproj** - the project file that is opened in Visual Studio, which holds the description of the project and all its related configuration and directives information.
- COBOL source files - when you create a project, a skeleton COBOL source file `Program1.cbl` is added for most of the project templates.

In the *Name* folder:

- `..\bin` - this is the default location of build artefacts. With this folder are the subfolders `x86\Debug` that contains the executables or libraries, and `.idy` file for each of the project's COBOL source files.

The `.idy` files contain information required for debugging your application. When you use the Release build configuration, build output goes to a subfolder `x86\Release` and no `.idy` files are created.

Debug and Release are standard build configurations that you launch from the Visual Studio task bar. They use a different set of compiler directives as well as outputting different files. You can create your own build configurations by clicking **Build > Configuration Manager** and choosing **New** from the **Active solution configurations** drop-down list.

The `x86` folder exists because the default output platform is 32-bit. If you change this to be 64-bit, you will instead find your output in an `x64` folder.

- `..\obj` - this also has `x86\Debug` subfolders, and contains an `.obj` file for each source file, used in intermediate build stages. The `obj` folder also holds supporting information such as logs and file lists.



Note: The project file `.cblproj` is an msbuild file, much like a makefile but consisting of XML that you can extend and modify to customise your builds. You can use this directly from command line, as it uses the same build environment as the IDE, and behavior is identical. This means you can have a single source of configuration information that makes your build process easier to maintain.

If you open a command prompt and change to the *Location* folder, you can execute the `msbuild` command, without needing to specify the `.cblproj` file.

Finding your way around the IDE's features

- **Solutions and projects**

A solution is a container holding one or more projects that work together to create an application. The solution has the extension `.sln`. A COBOL project has the extension `.cblproj` and a C# project has the extension `.csproj`.

Solution Explorer shows the solution that is open and the projects therein.

You can use the project's properties pages to display a list of the files in your solution with file details like output file type and location, COBOL dialect, and the number of errors generated by the file. To display the properties, click **Project > Name Properties**.

- **COBOL editor**

The COBOL editor provides help such as column cut and paste, and background syntax checking, which underlines errors with red wavy lines (also known as "squiggles"), which you can then hover over to display details of the syntax error.

When you are editing, you can insert code snippets and navigate forward and backward quickly, and the **Find All References** option enables you to search for references of any COBOL data items, section and paragraph names in the solution.

You can customize the editor to display line numbers, adjust colorization, tabs, and margins, from the **Text Editor > Micro Focus COBOL > Advanced** page in **Tools > Options**.

When developing code, the editor provides IntelliSense that helps you write syntactically correct code and, in managed code, helps when you need to type more complicated constructs, such as the code to override the members that a class inherits from a base class or the code for implementing an interface.

The Smart TagLight Bulbs feature for implementing an interface helps complete incomplete interface declarations. A Smart TagLight Bulbs appears at the beginning of the declaration: click it and choose the missing member(s) of the inheriting interface.

When you encounter a `COPY` statement, or data item that is defined in a copybook, if you put your cursor on that code and press F12 the appropriate copybook opens in the editor at the relevant line. You can also do this by right-clicking the line and selecting **Show copybook name**.

- **Setting Compiler directives**

Many Compiler directives are set automatically by certain configuration options in the IDE, but you can explicitly add directives to your project. Right-click on the project in Solution Explorer and choose

Properties. In the COBOL tab, you can see directives that are set by the IDE in the **Build Settings** text box. Enter others in the **Additional Directives** text box as a space-separated list.

If you use a separate text file to manage your directives, you can reference this instead by entering the `USE"directives file"` directive. You should enter a relative path.

- **Build Tools, the Output Pane and the Error List**

Build configurations define how to build a project or solution. There are default configurations of Debug and Release for each project type, and you can create your own specific configurations.

The Output window shows the results of your build together with errors. You can double-click an error and navigate directly to the appropriate line in the source code. You can do the same from the Error List.

- **Debugging**

When you debug the application, you can step through the code, hover over a data item to see its value, and watch data item values in a variety of ways. You can specify breakpoints on a range of conditions, such as when an expression is true or changes, or when a line is hit a specified number of times.

In native code, you can set COBOL watchpoints on data items and watch for changes in the area of memory associated with the watchpoints. When the memory changes, the debugger breaks on the line that follows the line on which the data change occurred.

Also in native code, you can use the Memory window to watch the contents of the memory that is associated with data items or expressions.

Change the Defaults to Replicate Your Existing Project Structure

Change the location of source files

To add an existing COBOL source file to your project, right-click the project in Solution Explorer and choose **Add > Existing item**. You can then browse to the sources you want to add.

- If you click **Add**, Visual COBOL makes a copy of the file, which it saves in the project folder. Any edits you make to this file do not get applied to the original.
- If you click **Add As Link**, a reference to the original file, rather than a copy of it, is added to the project in Solution Explorer. If you then open the file in Visual Studio, any edits are applied to the file in its original location.

You can also drag files from Windows Explorer and drop them into your project in Solution Explorer. This also makes a copy of the file and leaves the original in place.

To remove a file from your project, but not delete the file on disk (whether added as a link or not), right-click the file in Solution Explorer and choose **Exclude From Project**.

Change the location of built files

By default, built artefacts for the Debug configuration are created in the `..\Location\Solution\Name\bin\x86\Debug` folder.

You might want to change this, so that several developers can save built items in the same folder for example. To do this, right-click the project in Solution Explorer and choose **Properties**. In the COBOL tab, change the value of the **Output path** field to the preferred folder. (We recommend you always use relative paths when entering this value.) When the project builds, the output files will be saved in this folder, and the folder created if it doesn't already exist.

To change the output path for the Release configuration, select **Release Configuration** in the COBOL property page and change the value of the output path.

Change the type of built files

The default output and target types when you create a project depend on the project type. You can change these settings on the project **Properties** page. Use the following table to show the default output and target types for each project and the possible changes once the project has been created :

Project type	Output type	Target type	Possible output types	Possible target type	
Native	Console application	.exe	single	.dll, .exe	single, multi
	Windows application	.exe	single	.dll, .exe	single, multi
	Link library	.dll	single	.dll, .exe	single, multi
	Enterprise Server application	.dll	multi	.dll, .exe	single, multi
	INT/GNT application	.int	multi	.int, .gnt	multi
Managed	Console application	.exe	single	.dll, .exe	single
	Windows application	.exe	single	.dll, .exe	single
	Link library	.dll	single	.dll, .exe	single
	Procedural multi-output project	.dll	multi	.dll, .exe	multi

Best Practice in Visual COBOL Development

Break down large projects

Projects with a large number of source files and build artefacts can be hard to navigate and slow to build. If you find this the case, we recommend that you review the contents of large projects and split them into separate projects (and possible separate solutions) in which you group items that are logically related. These projects can still be built in the same output folder if required.

For example:

- If you have different versions of a product for different customers, keep common source in one project and a separate project for each customer. You could also have a master solution into which you add projects from other solutions by right-clicking a solution and selecting **Add > Existing Project**.
- If you have core code that is rarely changed or recompiled, keep that in one project and have separate projects for those areas that change regularly.

Referencing common sources

To avoid repetition and reduce maintenance effort, you should consider keeping all your Compiler directive settings in a directives file and reference this file in each project. Similarly you should keep copybooks in a single project and add this project as a dependency to your COBOL projects.

If using managed code and multiple projects, use project references rather than file references.

Create templates

After creating and configuring a project, you can save the settings as a template that can be reused and distributed to other users. It can be added to the list of project types available when clicking **File > New > Project > COBOL**.

To create a template of the open project, click **File > Export Template** and follow the steps explained in the Export Template Wizard.

Use relative paths

Keep source relative to a base path and avoid full paths so that code is portable and easy to use with source control systems. You should also avoid using network shares or drives.

Modernize Your Applications and Processes

Following industry standard development practises

Many source code control systems and Agile tools can be integrated into the Visual Studio IDE.

You should also consider using continuous integration, which involves the automatic building and testing of an application after a change occurs to the source code. This method traps errors sooner in the development life cycle and can greatly improve efficiency and reduce costs.

Interface modernization

Visual COBOL enables you to use Visual Studio's built-in design tools to create more intuitive user interfaces. By wrapping existing procedural COBOL in an wrapper class you can integrate your code into Windows Forms (WinForms) and Windows Presentation Foundation (WPF) technology, and WebForms for ASP.NET browser-based applications.

Multi-user applications

Visual COBOL includes a Run Unit API to enable multiple users to simultaneously use an application based on COBOL code that was designed originally for a single user.

Developing Web-based applications

You can use Visual COBOL to migrate existing, core applications to a service oriented architecture as Web services, and deploy them using Micro Focus COBOL Server and Enterprise Server, so that you can develop COBOL-based software components to be invoked across the Web.

You can do this by creating an Enterprise Server application

Developing .NET applications

Both new and existing COBOL can be compiled as .NET managed code. This enables you to:

- Reuse existing COBOL business logic and data access across the .NET environment
- Access .NET Framework classes and features from COBOL applications including Windows Forms and Web Forms
- Create and extend composite applications consisting of COBOL, C#, VB.NET, C++ and ASP.NET
- Reuse and extend Open ESQL applications

Both procedural and OO COBOL are supported within the .NET framework. OO COBOL classes can inherit classes written in other Microsoft .NET languages and vice versa.

The managed COBOL syntax includes many extensions to the COBOL language to support .NET features; for example, the TRY ... CATCH syntax to enable exception handling in COBOL.

There are also certain directives that help integrate your managed COBOL with other languages in the .NET environment. For example, you can now expose the Linkage section and entry points in your COBOL to other managed languages by compiling with the ILSMARTLINKAGE directive.

Modernizing Dialog System applications

Visual COBOL provides the following support for Dialog System applications:

- Dialog System run-time system and run-time components.

- Panels V2.
- Dialog System painter.
- GUI class library and OLE class library. These libraries are needed if you migrate an existing Dialog System application that was extended using those libraries.

Projects for building the GUI and OLE class libraries from source are also supplied. Additionally, a project file for the Base class library was added in Visual COBOL 2.0.

- Visual Studio plug-in to associate screensets in Visual Studio with Dialog System. Double-clicking a screenset in Solution Explorer in Visual COBOL starts the Dialog System painter.
- Sample applications demonstrating a range of modernization techniques.
- Supporting documentation in this Help explaining the significant elements of the sample code.

You can modernize Dialog System applications within Visual COBOL. You migrate an application to Visual COBOL and from there you can run the application without change, or modernize it over time.

Modernization techniques include:

- A Windows Forms form replacing a Dialog System dialog, where the form can contain .NET controls. See the Customer + .NET WinForm sample `CustomerWinForm.sln`.
- A Windows Forms control wrapped as an ActiveX control and used on a Dialog System dialog. See the Customer + .NET GridView User Control sample `custgrid.sln`.
- A WPF user control hosted by a Windows Forms user control, which is then exposed as ActiveX ready for use by Dialog System. See the Customer + .NET WPF GridView User Control sample `CustGridWPF.sln`
- A .NET managed code application interacting with Dialog System as native COBOL `.dll`. See the Managed Customer sample `ManagedCustomer.sln`.

Data File Tools

Visual COBOL comes with two versions of the Data File Tools utility: Data File Tools and Classic Data File Tools.

The Classic Data File Tools is the utility that was previously available in Net Express. It includes the Data File Converter, Data File Editor, and the Record Layout Editor. This utility is only available on Windows.

Data File Tools is a new version of the utility and comprises the Data File Editor and the Structure File Editor. This utility is available on both Windows and UNIX.

Procedural COBOL Compared with Managed COBOL

Procedural COBOL is regular COBOL without any of the new syntax that has been added for .NET and JVM. This is the COBOL that will have been used to write Net Express, Server Express and Mainframe Express applications, and it is still actively supported today.

You can compile to native or (in most cases) managed code. The core COBOL syntax is supported in managed code. However, there are some features that are not supported (for example Panels V2, Dialog System and ACUCOBOL-GT). This means that you can take most existing COBOL applications and recompile to create managed applications.

Managed COBOL

Managed COBOL is the collective term for .NET COBOL and JVM COBOL.

Managed COBOL is COBOL with extensions to support the .NET and JVM frameworks. It offers OO syntax support and syntax to allow access to the available class libraries.

When you compile managed COBOL the compiler generates managed code: `.il` for the .NET framework.

Managed Code and Native Code

You can compile your COBOL program to managed code using the `ilgen` compiler directive. From within the IDE this happens automatically if you are using a managed COBOL project.

The compiler has now created an intermediate language (`.i1`).

COBOL and all other .NET languages (for example C# and VB) compile to this format, which makes mixed language applications easy to write.

You can also create native code applications. In Visual Studio there are native COBOL project templates.

The compiler generates `.exe/.dlls` as the result of a native compilation.

The native COBOL application has to call the appropriate management services available for the operating system, whereas a managed application can take advantage of the management services provided by the run time such as exception handling, garbage collection, and thread management.

Run Time

The intermediate language (`.i1` files) can be deployed to a Windows platform running Microsoft's Common Language Runtime (CLR) for execution.

All programs written for the .NET Framework are executed by the Microsoft's Common Language Runtime (CLR) which makes mixed language application programming seamless.

The CLR's just-in-time (JIT) compiler compiles the `.i1` into code native to the operating system. The CLR provides additional services including memory management, exception handling, garbage collection and thread management.

Developing Native and Managed Applications

You use the IDE to develop, compile and debug both native and managed applications. You can write new COBOL code or you can recompile existing COBOL applications to managed or native code, potentially without any code changes.

You can deploy and further debug the application under the run-time system provided by COBOL Server.

.NET COBOL applications are deployed to Windows platforms running Microsoft's CLR.