Upgrading to Visual COBOL 2.0 for Eclipse

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

2012-05-21

Contents

Upgrading to Visual COBOL 2.0 for Eclipse

This guide provides information on upgrading applications from earlier Micro Focus COBOL development systems to Visual COBOL for Eclipse. It highlights the differences between the old and new products, and offers solutions on how to keep your application working in the same way as before. The guide also introduces the new concepts and features of the Integrated Development Environment.

Note:

- This documentation uses the name Visual COBOL to refer to Visual COBOL for Visual Studio and Visual COBOL for Eclipse. The full product names are used only when it is necessary to differentiate between the two products.

Benefits of Upgrading

You get a number of important benefits by upgrading to Visual COBOL from earlier Micro Focus development systems or other COBOL systems, such as RM/COBOL and extend® (ACUCOBOL-GT).

Visual COBOL uses a proven industry Integrated Development Environment that supports thousands of clients for developing and deploying critical business applications. Visual COBOL enables unified, collaborative, and cost-effective development through rich, industry-standard tooling and at the same time it helps minimize skills shortages, expands market reach and accelerates time-to-delivery to meet today's agile business requirements.

Visual COBOL helps improve developer productivity and application quality, helps reach new markets and audiences, and makes COBOL equivalent to all other contemporary languages.

With the capabilities of the new IDE, you can reach new platforms with little or no change and deploy applications faster across 50 platforms including NET, Azure, and JVM.

Licensing Changes

For a number of years Micro Focus used the Micro Focus License Management System on Windows and Micro Focus License System Administration on UNIX for Net Express and Server Express.

Micro Focus now uses a standard industry technology for license management, Sentinel RMS from SafeNet. New product releases use Sentinel RMS, as do updates to existing products.

For more on licensing, see *Licensing* in the Visual COBOL help.

Resolving Conflicts Between Reserved Keywords and Data Item Names

Micro Focus continues to expand the list of reserved COBOL words by adding new keywords to it as part of new levels of the COBOL language. Each Micro Focus release corresponds to a particular level. You can use the MFLEVEL Compiler directive to enable Micro Focus-specific reserved words in your code and change the behavior of certain features to be compatible with a specific level of the language.

If you use Visual COBOL to compile applications created with an older Micro Focus product, and these applications use data names that are now reserved keywords in Visual COBOL, you receive a COBOL syntax error COBCH0666 ("Reserved word used as data name or unknown data description qualifier"). To work around this issue and continue using some of the reserved words as data names in your source code, you can either:

- use the REMOVE Compiler directive to remove individual keywords from the reserved words list

- set the MFLEVEL Compiler directive to a lower level which corresponds to the level your applications are at (see the information about MFLEVEL of some Micro Focus products further down this section). This removes all reserved keywords which have been added for levels above that level from the reserved words list.

You can set both directives from the command line, in your source code, or in the **Additional Directives** field in the project's COBOL properties.

Setting directives from the command line

To use REMOVE from aVisual COBOL command line, 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 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"
```

Setting directives in the IDE

To set either one of the directives in the project's properties:

1. In the IDE, click **Project** > **Properties** > **Micro Focus COBOL** > **Project Settings** > **COBOL**.
2. Type MFLEVEL"*nn*" or REMOVE "*ReservedWord*" in the **Additional directives** field.
3. Click **Apply** and then **OK**.

MFLEVEL of some Micro Focus product releases and reserved words added for them

These are the keywords that have been added to the reserved words list for some of the more recent Micro Focus products:

- Visual COBOL R4 (MFLEVEL"16"):

  ATTRIBUTES
  ENCODING
  NAMESPACE
  NAMESPACE-
  VALIDATING
  XML-
  XML-SCHEMA
- Net Express and Server Express versions 6.0 WrapPack 2 and 5.1 WrapPack 5 (MFLEVEL "15"):

  DATA-POINTER
  OBJECT-REFERENCE
- Net Express 6.0 and Server Express 6.0 (MFLEVEL "14"):

  BIT
  BOOLEAN
  GROUP-USAGE

For more information on the MFLEVEL Compiler directive and the keywords used by the different product versions, read the following topics in the product Help:

General Reference > COBOL Language Reference > Part 4: Appendices > Reserved Words
General Reference > Compiler Directives > Compiler Directives Alphabetic List > MF, MFLEVEL

Recompile All Source Code

Application executables that were compiled using Net Express, Server Express, RM/COBOL or extend® (ACUCOBOL-GT) 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. The error might be similar to this, on UNIX:

```
ld.so.1: rts32: fatal: libcobrts.so.2: open failed: No such file or directory
Killed
```

You can recompile from the IDE or the command line.

Upgrading from ACUCOBOL-GT

There are conceptual and behavioral differences between Visual COBOL and ACUCOBOL-GT, part of the Micro Focus **extend®** product family, and these differences can affect the way you upgrade existing applications to Visual COBOL.

Note: At the time of publishing this guide, Visual COBOL is not fully compatible with ACUCOBOL-GT. We recommend native code deployment only.

Micro Focus will continue to improve the compatibility between the two products with the future releases of Visual COBOL.

Compiling Your ACUCOBOL-GT Applications in Visual COBOL

You can use the standard Visual COBOL tools and the Visual COBOL command line to compile and debug ACUCOBOL-GT applications to the runnable formats that Visual COBOL supports.

Enabling compatibility with ACUCOBOL-GT in Visual COBOL

Compatibility with ACUCOBOL-GT's language extensions and data files is not turned on by default in Visual COBOL so you must compile your code with the ACU Compiler directive. You can set this directive in your source code directly, through the project properties, or at the command line.

To set the ACU directive in your source code, type the following at the beginning of your program with the $ in the indicator area, :

```
$set ACU
```

If you use the Visual COBOL command prompt, you can compile as follows:

```
cobol myprogram.cbl acu obj();
```

If you use the IDE to edit and build the code, you must set the ACU directive in the project properties before you compile:

1. In the IDE, click **Project** > **Properties** > **Micro Focus COBOL** > **Project Settings** > **COBOL**.
2. Type ACU in the **Additional directives** field.
3. Click **Apply** and then **OK**.

Specifying ACUCOBOL-GT Compiler Options

The COBOL Compiler in Visual COBOL supports many of the Compiler options available with the ACUCOBOL-GT (ACU) Compiler. To set these options, you need to compile with the ACUOPT directive (which automatically sets the ACU directive). For example:

```
cobol myprogram.cbl acuopt(-Dd31 -Gd) obj();
```

You can also use the ccbl.exe utility, an interface to the Micro Focus Compiler that accepts the same options and parameters as the ACU Compiler, to compile your code. It automatically sets the ACU directive.

ccbl produces .int by default:

```
ccbl myprogram.cbl
```

To compile to .gnt, you need to compile with the native code option, -n:

```
ccbl -n myprogram.cbl
```

For more details, see *Compatibility with ACUCOBOL-GT* in the Visual COBOL Help.

Accessing Vision Indexed Files from Visual COBOL

You can access Vision Files directly from native COBOL.

Accessing Vision files from native COBOL

There are two ways to provide access to Vision files in native COBOL:

- Compile your applications with the CALLFH"ACUFH" Compiler directive set and then link `acufh.lib` to the application. This enables you to use Vision files only.
- Use FHREDIR, the dynamic redirection capability of the File Handler. This enables you to mix Vision and Micro Focus data files.

To compile from a COBOL command environment with CALLFH"ACUFH" and link to `acufh.lib` to your native application, execute:

```
cobol myprogram.cbl callfh(acufh) obj();

cblink myprogram.obj acufh.lib
```

To use the IDE to link `acufh.lib` to your native application:

1. In the IDE, click **Project** > **Properties** > **Micro Focus COBOL** > **Build Configurations** > **Link** > **Additional Link Files**.
2. Browse to the location of the `acufh.lib` file and add it to your project:
   - On a 32-bit system, the default location is `C:\Program Files\Micro Focus\Vision 2.0\lib\acufh.lib`
   - On a 64-bit system, when building a 32-bit executable, include `C:\Program Files (x86)\Micro Focus\Vision 2.0\lib\acufh.lib`
   - On a 64-bit system, when building a 64-bit executable, include `C:\Program Files (x86)\Micro Focus\Vision 2.0\lib64\acufh.lib`

Note:

- If you do not link `acufh.lib` to the application, you receive an error - `Unresolved external symbol _ACUFH`.
- If you compile your applications from the command line to .int code, you do not have to link to the `acufh.lib` library – the run-time system finds it automatically.

Accessing Vision files from managed COBOL

In Visual COBOL, support for Vision data files is only available for native COBOL applications. You cannot access them directly in managed code. Support for Vision files in managed COBOL is planned for future releases of Visual COBOL.

We recommend the following solutions, in order of preference:

- Convert your Vision data files to Micro Focus data files. To do this, use the data migration tool, ACU2MFDataMigration.exe, which is available in the folder where the product samples are installed.
- Create a COBOL application which builds to a native .dll to access the files directly, and then call the .dll from your managed code.

After converting a Vision data file to work with managed code, you may find that the managed code file handler is much slower than the native one. If so, set up the managed file handler so that it uses Fileshare instead of the external file handler.

Library Routines

To use the ACUCOBOL-GT standard library routines in Visual COBOL, you need to compile your applications with the ACU Compiler directive which enables ACUCOBOL-GT compatibility in Visual COBOL.

The following standard library routines are available, but for native COBOL only:

C$CALLEDBY
C$CALLERR
C$CHDIR
C$MAKEDIR
C$MEMCPY
C$MYFILE
C$NARG
C$PARAMSIZE
C$RERR
M$ALLOC
M$FREE
M$COPY
M$FILL
M$GET
M$PUT
WIN$VERSION

For more information on each library routine, see *ACUCOBOL-GT Library Routines* in the product Help.

Supported Features

The following ACUCOBOL-GT functionality is supported in Visual COBOL:

- ACU4GL - this is now known as Micro Focus Database Connectors™
- Vision, ACUCOBOL-GT's native indexed file system, is supported. This enables you to use your existing data files. The Vision-related utilities acusort.exe, logutl32.exe, vio32.exe and vutil32.exe, are supported.
- Micro Focus XDBC™

Restrictions and Unsupported Features

The following is not an exhaustive list of the restrictions of using ACUCOBOL-GT in Visual COBOL. In most cases, if your code includes ACUCOBOL-GT features not supported by Visual COBOL, you will receive a Compiler error.

- The ACUCOBOL-GT multi-threading model is not supported.
- The ACUCOBOL-GT configuration file and configuration variables are not supported. Visual COBOL uses different configuration files and variables. You need to review your existing ACUCOBOL-GT configuration to determine which settings are relevant for use with Visual COBOL and which settings have Visual COBOL equivalents. Any variables associated with Vision files, for example, can be set in a configuration file (or set in the environment, either from the command line or within the COBOL program). Some ACU configuration variables are not necessary or applicable in Micro Focus COBOL (for example, PERFORM_STACK), and the functionality of others is covered by the Micro Focus compile and run-time options (for example, A_CHECKDIV). See the product Help for more information.
- The ACUCOBOL-GT Thin Client technology is not supported.
- The Graphical Technology (GT) is not supported.
- Visual COBOL and extend® differ in their support for some of the Screen Description phrases. In Visual COBOL, the following phrases of the Screen Description entry are not supported and should be removed from your programs:

  AFTER
  BEFORE
  EXCEPTION
- Moving ACUCOBOL-GT applications to managed COBOL is not supported yet. Managed COBOL does not support:
  - the ACU numeric sign encoding schemes - sign(ascii), sign(ebcdic), sign(acu), sign(mbp), sign(ncr), sign(realia) and sign(vax)
  - some ACU data types such as comp-3, comp-6, comp-4
  - size error checking
  - truncation
- ACUFH does not currently support assigning files to pipes - for example:

  ```
  select test-file assign to "-P %TMP% cmd /c dir *.* > %TMP%"
  ```

  ```
  select test-file assign to "-P ls *"
  ```

  To work around this issue:

  1. Create a subprogram which does not use a CALLFH"ACUFH" statement and which handles the required pipes. The syntax for assigning file to pipes is different in the Visual COBOL File Handler:

     ```
     select test-file assign to "<cmd /c dir *.*
     ```

     ```
     select test-file assign to "<ls *"
     ```
  2. Call the subprogram from the program using CALLFH"ACUFH".

  For more information, read *Programming > File Handling > File Handling Guide > Filenames > Setting Up Pipes* in the product Help.

Upgrading from Earlier Micro Focus Products

You can upgrade COBOL applications that were developed in Net Expressand Server Express to Visual COBOL. The majority of the existing applications will continue to run in Visual COBOL without the need to change their code.

This guide lists the differences between Net Express, Server Express and Visual COBOL in the following areas:

Compiling and building	Having created a project in Visual COBOL, you can either use the IDE or the command line to build.
Run-time systems	There are some differences between the run-time systems supplied with Visual COBOL and those supplied with Net Express and Server Express. This, however, will not affect your existing applications and they will continue to run under Visual COBOL - you only need to recompile the applications from the source code with Visual COBOL.
Run-time system technologies	Some technologies behave differently and require some upgrade work.
Restrictions and unsupported features	Some features of Net Express and Server Express are not available in Visual COBOL. However, there are alternative techniques for many of these features.
Editing and debugging	Much of the Net Express and Server Express functionality for editing and debugging is available in Visual COBOL, but sometimes with a different name and with a slightly different behavior. In addition there are some new features such as background parsing, which highlights errors as you type and code completion techniques that provide easy access to language elements, enabling you to select and insert them simply.
Eclipse integration	Visual COBOL is integrated with the Eclipse IDE. The development environment provides all the functionality to manage projects and debug applications. COBOL applications previously built in Net Express or Server Express can be developed and run within the Eclipse IDE.

Summary of Differences

The majority of the applications created with Net Express or Server Express will continue to work in Visual COBOL without any changes. However, there are some differences between these development systems you should consider when you upgrade to Visual COBOL.

Compiling and Building Differences

There are several aspects of compiling and building applications that behave differently in Visual COBOL. You might need to change the project properties and update some of the Compiler directives and settings that you previously used.

Output File Formats **on page 14**	The preferred executable file formats with Visual COBOL are .dll, .so and .exe. The .int and .gnt file formats are still supported by the Compiler and the debugger but cannot be created directly by the IDE.
Compiler Directives **on page 14**	When you upgrade your source code to Visual COBOL some Compiler directives that were specifically designed for 16-bit systems now produce an error on compilation because they are no longer relevant. You should remove them from your code and directives files before you compile.
Linking **on page 15**	The static run-time system and the single-threaded run-time system on Windows are no longer required and they are not shipped with Visual COBOL. Applications built with Visual COBOL are now linked to the shared or dynamic run-time systems. On UNIX, you can link to the single-threaded or multi-threaded shared or dynamic run-time system.

***Called Programs and Dependencies* on page 15**	At run time, called programs are found in the same way as before. However, there are some new ways to set COBPATH and copy files into a common folder.
***File Handler* on page 16**	The File Handler .obj files are not available in Visual COBOL. Visual COBOL uses the File handler packaged in the `mffh.dll` file instead.
***Makefile Conversion* on page 16**	You cannot use existing makefiles from Visual COBOL for Eclipse, although where they invoke commands supported by Visual COBOL they will continue to work as before.
OpenESQL Assistant	The OpenESQL Assistant data source names (DSNs) in Visual COBOL must be configured as ODBC or ADO.NET DSNs.
***SQL Compiler Directive Options* on page 16**	When you upgrade your SQL appplications to Visual COBOL, some applications could require additional SQL compiler directive options to avoid compiler errors.
***XML PARSE Statement* on page 16**	In Net Express, the default setting for the XMLPARSE Compiler directive is COMPAT, which causes the XML PARSE statement to return information and events for IBM Enterprise COBOL Version 3. In Visual COBOL, the default is XMLPARSE(XMLSS), which returns information and events for IBM Enterprise COBOL Version 4.

Run-Time System Differences

There are some differences between the run-time systems supplied with Visual COBOL and those supplied with Net Express, Server Express and Mainframe Express. These, however, do not affect your existing applications if you recompile them from the source code in Visual COBOL.

***OpenESQL* on page 16**	Visual COBOL sets the BEHAVIOR SQL Compiler directive option to MAINFRAME by default to provide optimal performance. To revert to the default behavior exhibited in Net Express, set the BEHAVIOR directive to UNOPTIMIZED.
***Single-Threaded Run-Time System* on page 16**	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. On UNIX, the single-threaded run-time system is available, so that applications can link with third-party code.
***Static-Linked Run-Time System* on page 16**	The static-linked run-time system is not available in Visual COBOL. Instead, you now link native code to the shared or dynamic run-time system. This has no effect on your existing applications.
***Setting the Environment on UNIX* on page 17**	You use the cobsetenv script to set your COBOL environment on UNIX.
***Visual COBOL Co-existing with Earlier Micro Focus Products* on page 17**	Some additional configuration is required to ensure Visual COBOL and Net Express or Studio Enterprise Edition work properly when installed on the same machine.

Restrictions and Unsupported Features

Some features in earlier Micro Focus products are not available in Visual COBOL. However there are alternative techniques for many of these features.

***Character-Mode Dialog System* on page 17**	Support for creating character-based user interfaces for applications that run in character environments is available for Visual COBOL if you install the Character-Mode Dialog System AddPack, distributed for free through the *Micro Focus SupportLine Web site*.

COBOL Services as Java and Web Services **on page 17**	COBOL services, such as Java interfaces and Web Services, created with the Interface Mapping Toolkit in Net Express or Server Express run only under Enterprise Server within Micro Focus Server. They do not run under COBOL Server.
CSBIND **on page 17**	CSBIND is a service package in Net Express, Server Express and Application Server that supports client/server COBOL applications. It is not available in Visual COBOL.
DBMS Preprocessors **on page 17**	Earlier Micro Focus products supported DBMS preprocessor versions that are not supported in Visual COBOL. For a list of currently supported DBMS preprocessors, see the *Database Access Support with Native COBOL* topic.
Dialog System **on page 17**	Dialog System applications are not supported in Eclipse but you can upgrade the non-GUI components of an application to Eclipse, and then recreate the GUI components using the GUI tools in Eclipse.
Enterprise Server **on page 17**	Enterprise Server and Server for SOA provide an execution environment for COBOL services and COBOL application programs, including mainframe support for CICS, JCL, and IMS. They are not available in Visual COBOL or COBOL Server.
Form Designer **on page 18**	Form Designer is the Net Express tool for creating user interfaces for CGI-based Internet and intranet applications. Form Designer and the HTML page wizard are not available in Visual COBOL.
FSView **on page 18**	FSView is a utility for administering Fileshare servers. The FSView GUI is not supported in Visual COBOL.
GNT Analyzer **on page 18**	GNT Analyzer is not available in Visual COBOL. It has been replaced by *Test Coverage*.
Host Compatibility Option (HCO) **on page 18**	Host Compatibility Option (HCO) is not supported in Visual COBOL.
Interface Mapping Toolkit **on page 18**	The Interface Mapping Toolkit is not supported in Visual COBOL.
INTLEVEL Support **on page 18**	The INTLEVEL directive is rejected by the Compiler in Visual COBOL.
J2EE Application Servers **on page 18**	J2EE Application Servers are not supported by Visual COBOL.
NSAPI **on page 18**	There is no support for NSAPI in Visual COBOL.
Online Help System **on page 18**	Net Express provided the Online Help System for creating online help from character-based applications, and displaying it on screen. It is not available in Visual COBOL and the Online Help System information file type (.HNF) is not supported.
OO Class and Method Wizards **on page 18**	The OO Class and Methods wizards are not available in Visual COBOL. However, the run-time components for the base and COM OO class libraries are available.
OpenESQL **on page 18**	In both Net Express and Studio Enterprise Edition, support is provided for Oracle OCI in OpenESQL. Visual COBOL does not support Oracle OCI in OpenESQL.
OpenESQL Assistant **on page 18**	The OpenESQL Assistant is not available in Visual COBOL for Eclipse.

Secure Sockets Layer (SSL) **on page 18**	Secure Sockets Layer (SSL) is a standard mechanism for sending and receiving electronic communications in encrypted form. It is not currently supported in Visual COBOL.
Solo Web Server **on page 19**	The Solo Web server in Net Express enabled you to debug CGI-based Internet applications on the same machine you used to develop them. It is not available in Visual COBOL.
SQL Option for DB2 **on page 19**	SQL Option for DB2, also known as XDB, is not supported in Visual COBOL.
Type Library Assistant **on page 19**	Type Library Assistant is not included in Visual COBOL but the run-time components for the COM and the OO COBOL libraries are still available.
TX Series	The IBM TX Series product used to interface with Websphere in Net Express is not supported in Visual COBOL.
UNIX Publish **on page 19**	The UNIX Publish feature is superseded by the remote development functionality in Visual COBOL for Eclipse. You use Visual COBOL Development Hub, a remote development server to host your source code and you use the Eclipse IDE on your local machine as the development interface.

Run-Time Technology Differences

Some technologies behave differently in Visual COBOL and this might affect how you upgrade existing applications.

COM Interop **on page 19**	The tools to help create COM objects are not supplied with Visual COBOL. However, the COM run-time components are supplied, so that COM is supported and your applications can interoperate with existing COM objects.
File Handling **on page 19**	The way you integrate your own security modules into Fileshare has changed. Also, the FILEMAXSIZE setting is different for Visual COBOL and for Net Express and Server Express .
Java and COBOL **on page 20**	The `cobsje` script is not available in Visual COBOL for Eclipse on UNIX. Visual COBOL uses the COBOL run-time system to load JVM based on LIBPATH, LD_LIBRARY_PATH, SHLIB_PATH, JAVA_HOME etc.
Test Coverage **on page 20**	Visual COBOL supports Test Coverage from the command line only.

Editing and Debugging Differences

Much of the edit and debug functionality in Net Expressand Server Express is available in Visual COBOL, but some of it has a different name or slightly different behavior. In addition there are some new features such as background parsing.

Data Tools **on page 20**	The Net Express Data Tools are available as an AddPack for Visual COBOL for Eclipse on Windows, but not on UNIX.
Debugging Native Object-Oriented COBOL **on page 20**	In Net Express you can examine an object while debugging OO COBOL and display the class that defined the object and also other objects derived from that class. In Visual COBOL, you can also view the class information of native OO COBOL but not while debugging.
Mixed Language Debugging **on page 20**	With Net Express you can debug mixed language applications. Visual COBOL does not fully support mixed language debugging of native code.

Program Breakpoints **on page 21**	Program breakpoints are breakpoints that stop execution each time a specified program or entry point within the program is called. They are supported in Visual COBOL.
Remote Debugging **on page 21**	The Net Express animserv utility used for debugging programs remotely has been replaced by `cobdebugremote` (Windows) or `cobdebugremote32` (UNIX) (or `cobdebugremote64` when debugging 64-bit processes) in Visual COBOL.
Source Pool View **on page 21**	The source pool view in Net Express showed all source files available in the project directory, regardless of whether or not they are used in the current build type. This view is not available in Visual COBOL.

Compiling and Building Differences

There are several aspects of compiling and building applications that behave differently in Visual COBOL. You might need to change the project properties and update some of the Compiler directives and settings that you previously used.

Output File Formats

Preferred file formats - .exe and .dll

The preferred executable file formats with Visual COBOL are .dll, .so and .exe. The .int and .gnt file formats are still supported by the Compiler and the debugger but cannot be created directly by the IDE.

Building to multiple output files

Each project compiles into a single file (.dll, .so or .exe), or to multiple files of the same file type with one output file for each source file.

Instead of an .lbr file, which contained a collection of .int and .gnt files on Windows, you now use a .dll as the container for application components.

Your application can consist of multiple projects, each one building a single output file, with the projects linked together so that they can access each other.

1. Click **File > New > Other**.
2. Expand **General** and click **File**.
3. Enter the name to use for the linked resource. You can enter an alias.
4. Click **Advanced**.
5. Select **Link to file in the file system** and browse to the file to link to. You can also click **Variables** and define a path variable, so that you can link to files using a relative path.

Compiler Directives

When you upgrade your source code to Visual COBOL some Compiler directives that were specifically designed for 16-bit systems now produce an error on compilation because they are no longer relevant.

The following Compiler directives are no longer relevant and we recommend that you remove them from your code and directives files before you compile:

01SHUFFLE	FIXING	REGPARM
64KPARA	FLAG-CHIP	SEGCROSS
64KSECT	MASM	SEGSIZE
AUXOPT	MODEL	SIGNCOMPARE
CHIP	OPTSIZE	SMALLDD
DATALIT	OPTSPEED	TABLESEGCROSS

EANIM	PARAS	TRICKLECHECK
EXPANDDATA	PROTMODE	

Linking

The static run-time system and the single-threaded run-time system on Windows are no longer required and they are not shipped with Visual COBOL. Applications built with Visual COBOL are now linked to the shared or dynamic run-time systems. On UNIX, you can link to the single-threaded or multi-threaded shared or dynamic run-time system.

Linking from the command line

You can link applications from the Visual COBOL command prompt with the cbllink or cblnames commands. For example, to produce an .exe file, use:

```
cbllink myprogram.cbl
```

To compile and link your code to produce a .dll file, use:

```
cbllink -d myprogram.cbl
```

With these commands, the single-threaded and static-linking options are automatically mapped onto the multi-threaded and shared run-time systems respectively.

For more see *Command Line Reference* in the product help.

Linking from the IDE

To specify what to link:

1. Click **Project** > **Properties**.
2. Expand **Micro Focus COBOL** > **Build Configurations**.
3. Click **Link** and specify your link settings.

Called Programs and Dependencies

At run time, called programs are found in the same way as before. However, there are some new ways to set COBPATH and copy files into a common folder.

To build the called programs

When you build the called programs into a .dll or .so file, you can set a property to output the built file into the same folder as the main application executable. To do this the called programs and the application main program must be in the same project. To set this:

1. Create a project that contains the programs that are called.
2. In the project properties, expand **Micro Focus COBOL** > **Build Configurations** and click **COBOL**.
3. In **Target Settings**, select **Single Native Library File**.
4. Set the **Output Path** to the subfolder where the main application executable is output.
5. If you want to debug the .dll file together with the application, click **Override COBOL project settings** and check **Compile for debugging**.
6. Build the project.

To set the COBPATH environment variable

Add the COBPATH environment variable to the run-time configuration, as follows:

1. In the properties of the calling application project, expand **Micro Focus COBOL** > **Run-time Configuration** > **Environment Variables**.
2. Add the COBPATH variable and specify the full path to your called files.

File Handler

The File Handler .obj files are not available in Visual COBOL. Visual COBOL uses the File handler packaged in the `mffh.dll` file instead.

If the application you are upgrading from Net Express used the File Handler .obj files, when you link your application in Visual COBOL the linker will emit a warning. The application will continue to operate as before provided that you supply the `mffh.dll` file with it.

Makefile Conversion

You cannot use existing makefiles from Visual COBOL for Eclipse, although where they invoke commands supported by Visual COBOL they will continue to work as before.

It is not possible to use or convert existing makefiles and Visual Studio MSBuild files.

SQL Compiler Directive Options

If you get errors in Visual COBOL when compiling an object application that was created in Net Express or Studio Enterprise Edition, recompile specifying the GEN-CLASS-VAR SQL Compiler directive option in addition to other appropriate options.

XML PARSE Statement

In Net Express, the default setting for the XMLPARSE Compiler directive is COMPAT, which causes the XML PARSE statement to return information and events for IBM Enterprise COBOL Version 3. In Visual COBOL, the default is XMLPARSE(XMLSS), which returns information and events for IBM Enterprise COBOL Version 4.

To emulate the Net Express behavior in Visual COBOL, specify the XMLPARSE(COMPAT) Compiler directive option.

For a summary of the differences in event information between XMLPARSE(XMLSS) and XMLPARSE(COMPAT), see the *Special Registers* topic in your Visual COBOL documentation.

Run-time System Differences

There are some differences between the run-time systems supplied with Visual COBOL and those supplied with Net Express, Server Express and Mainframe Express. These, however, do not affect your existing applications if you recompile them from the source code in Visual COBOL.

The changes in the run-time system are described in the following sections.

OpenESQL

Visual COBOL sets the BEHAVIOR SQL Compiler directive option to MAINFRAME by default to provide optimal performance. To revert to the default behavior exhibited in Net Express, set the BEHAVIOR directive to UNOPTIMIZED.

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. On UNIX, the single-threaded run-time system is available, so that applications can link with third-party code.

Static-Linked Run-Time System

The static-linked run-time system is not available in Visual COBOL. Instead, you now link native code to the shared or dynamic run-time system. This has no effect on your existing applications.

See *Linking Native COBOL Code* in the product Help.

Setting the Environment on UNIX

Before you start developing applications with Visual COBOL for Eclipse on UNIX you need to set the environment. We recommend that you use the cobsetenv script to do this:

```
. /opt/microfocus/VisualCOBOL/bin/cobsetenv
```

Visual COBOL Co-existing with Earlier Micro Focus Products

If you have Visual COBOL and Net Express or Studio Enterprise Edition installed on the same machine, you sometimes receive a run-time system error if either the COBCONFIG or COBCONFIG_ environment variable is set when you run aVisual COBOL application the configuration file it refers to contains entries that are not valid for Visual COBOL.

To work around this issue, ensure that Visual COBOL is not running and then modify the configuration file by doing one of the following:

- If the invalid tunable is not needed by another application, remove it from the run-time configuration file.
- Add the following as the first line in the configuration file:

  ```
  set cobconfig_error_report=false
  ```
- Unset COBCONFIG (or COBCONFIG_) or set it to another configuration file that does not contain the invalid tunable for the particular session you are running in.

Restrictions and Unsupported Features

Some features in earlier Micro Focus products are not available in Visual COBOL. However there are alternative techniques for many of these features.

Character-Mode Dialog System

Support for creating character-based user interfaces for applications that run in character environments is available for Visual COBOL if you install the Character-Mode Dialog System AddPack, distributed for free through the *Micro Focus SupportLine Web site*.

COBOL Services as Java and Web Services

COBOL services, such as Java interfaces and Web Services, created with the Interface Mapping Toolkit in Net Express or Server Express run only under Enterprise Server within Micro Focus Server. They do not run under COBOL Server.

CSBIND

CSBIND is a service package in Net Express, Server Express and Application Server that supports client/server COBOL applications. It is not available in Visual COBOL.

DBMS Preprocessors

Earlier Micro Focus products supported DBMS preprocessor versions that are not supported in Visual COBOL. For a list of currently supported DBMS preprocessors, see the *Database Access Support with Native COBOL* topic.

Dialog System

Dialog System applications are not supported in Eclipse but you can upgrade the non-GUI components of an application to Eclipse, and then recreate the GUI components using the GUI tools in Eclipse.

Enterprise Server

Enterprise Server and Server for SOA provide an execution environment for COBOL services and COBOL application programs, including mainframe support for CICS, JCL, and IMS. They are not available in Visual COBOL or COBOL Server.

Form Designer
Form Designer is the Net Express tool for creating user interfaces for CGI-based Internet and intranet applications. Form Designer and the HTML page wizard are not available in Visual COBOL.

In Visual COBOL for Eclipse, you can install one of the available plug-ins to replicate these features.

FSView
FSView is a utility for administering Fileshare servers. The FSView GUI is not supported in Visual COBOL.

Visual COBOL provides all the FSView functions through the command-line utility `fsview`. For more information see *File Handling Reference* > *FSView* > *FSVIEW Command Line* in the product Help.

GNT Analyzer
GNT Analyzer is not available in Visual COBOL. It has been replaced by *Test Coverage*.

Host Compatibility Option (HCO)
Host Compatibility Option (HCO) is not supported in Visual COBOL.

Interface Mapping Toolkit
The Interface Mapping Toolkit is not supported in Visual COBOL.

INTLEVEL Support
The INTLEVEL directive is rejected by the Compiler in Visual COBOL.

An INTLEVEL of 1, 2, or 3 is no longer supported and causes compilation errors. Other values are reserved for internal use and should not be used.

J2EE Application Servers
J2EE Application Servers are not supported by Visual COBOL.

NSAPI
There is no support for NSAPI in Visual COBOL.

Online Help System
Net Express provided the Online Help System for creating online help from character-based applications, and displaying it on screen. It is not available in Visual COBOL and the Online Help System information file type (.HNF) is not supported.

OO Class and Method Wizards
The OO Class and Methods wizards are not available in Visual COBOL. However, the run-time components for the base and COM OO class libraries are available.

OpenESQL
In both Net Express and Studio Enterprise Edition, support is provided for Oracle OCI in OpenESQL. Visual COBOL does not support Oracle OCI in OpenESQL.

OpenESQL Assistant

Secure Sockets Layer (SSL)
Secure Sockets Layer (SSL) is a standard mechanism for sending and receiving electronic communications in encrypted form. It is not currently supported in Visual COBOL.

Solo Web Server
The Solo Web server in Net Express enabled you to debug CGI-based Internet applications on the same machine you used to develop them. It is not available in Visual COBOL.

In Visual COBOL, you need to use Apache2 or IIS servers for the CGI programs you create.

SQL Option for DB2
SQL Option for DB2, also known as XDB, is not supported in Visual COBOL.

Type Library Assistant
Type Library Assistant is not included in Visual COBOL but the run-time components for the COM and the OO COBOL libraries are still available.

TX Series
The IBM TX Series product used to interface with Websphere in Net Express is not supported in Visual COBOL.

UNIX Publish
The UNIX Publish feature is superseded by the remote development functionality in Visual COBOL for Eclipse. You use Visual COBOL Development Hub, a remote development server to host your source code and you use the Eclipse IDE on your local machine as the development interface.

With Visual COBOL, you store and develop your applications on a remote UNIX machine. You use a Windows or Linux-based Eclipse client to connect to the remote machine and you compile and debug the source code directly on it. Use Eclipse features such as the COBOL and Remote Systems Explorer perspectives within the IDE to background check syntax as well as build and debug your COBOL programs. As in other Micro Focus products, you use COBOL-specific projects to aid your everyday development activities within the IDE. This provides familiar functionality for editing, compiling, and debugging, complete with comprehensive COBOL Help.

For details, see *Using Eclipse for Remote COBOL Development* in the product Help.

Run-Time Technology Differences
Some technologies behave differently in Visual COBOL and this might affect how you upgrade existing applications.

COM Interop
The tools to help create COM objects are not supplied with Visual COBOL. However, the COM run-time components are supplied, so that COM is supported and your applications can interoperate with existing COM objects.

Documentation about COM Interoperability is available on the *Micro Focus SupportLine Web site* as part of the Net Express 5.1 documentation. See *Programming > COM and COBOL* in your product documentation.

File Handling
The way you integrate your own security modules into Fileshare has changed. Also, the FILEMAXSIZE setting is different for Visual COBOL and for Net Express and Server Express .

Using security modules

The way you integrate your own security modules (`fhrdrpwd`, `fsseclog` and `fssecopn`) into Fileshare has changed.

In Visual COBOL, you no longer relink Fileshare but you need to supply your own separate files, which are .dll files on Windows or shared objects on UNIX. For more information, see *Writing Your Own*

FHRdrPwd Module, *File Access Validation Module* and *Logon Validation Module* in the *File Handling* section of your product Help.

To use `fsseclog` and `fssecopn`, you need to link one or both of them into a `cobfssecurity.dll` or a shared object and place on the search path. Fileshare will issue a message indicating that it has loaded user security modules.

On UNIX, you no longer use `fsclose` to call Fileshare Manager. You need to use `cobfsclose` instead.

Sharing data files between applications built in Visual COBOL and others built using Net Express or Server Express

If you have applications that access the same data files, all those applications should be built with the same FILEMAXSIZE setting. However, applications built with Visual COBOL use a default setting of FILEMAXSIZE=8 while those built in Net Express or Server Express use FILEMAXSIZE=4.

In Visual COBOL you need to set the FILEMAXSIZE setting in the file handler configuration file (EXTFH.CFG). This ensures Net Express, Server Express, and Visual COBOL are all using the same setting and that programs running under the Net Express or Server Express run-time systems do not access the same files as programs running under the Visual COBOL run-time system.

Btrieve

Btrieve is the file handling system from Pervasive Software Inc. It is not supported in Visual COBOL.

Java and COBOL

The `cobsje` script is not available in Visual COBOL for Eclipse on UNIX. Visual COBOL uses the COBOL run-time system to load JVM based on LIBPATH, LD_LIBRARY_PATH, SHLIB_PATH, JAVA_HOME etc.

Test Coverage

Visual COBOL supports Test Coverage from the command line only.

Test Coverage replaces GNT Analyzer, which was available in Server Express.

Editing and Debugging Differences

Much of the edit and debug functionality in Net Expressand Server Express is available in Visual COBOL, but some of it has a different name or slightly different behavior. In addition there are some new features such as background parsing.

Data Tools

The Net Express Data Tools are available as an AddPack for Visual COBOL for Eclipse on Windows, but not on UNIX.

The Micro Focus Data File Tools AddPack includes the Data File Converter, Data File Editor, and Record Layout Editor.

You can download the Micro Focus Data File Tools AddPack from the *Micro Focus SupportLine site*.

Debugging Native Object-Oriented COBOL

In Net Express you can examine an object while debugging OO COBOL and display the class that defined the object and also other objects derived from that class. In Visual COBOL, you can also view the class information of native OO COBOL but not while debugging.

Mixed Language Debugging

With Net Express you can debug mixed language applications. Visual COBOL does not fully support mixed language debugging of native code.

To debug applications that contain programs in different languages, you need to debug the native COBOL and the non-COBOL code separately.

In Visual COBOL, you can debug mixed native COBOL and Java, however it involves a complicated setup. Mixed-language debugging of native COBOL and C is not supported.

Program Breakpoints

Program breakpoints are breakpoints that stop execution each time a specified program or entry point within the program is called. They are supported in Visual COBOL.

Remote Debugging

The Net Express animserv utility used for debugging programs remotely has been replaced by `cobdebugremote` (Windows) or `cobdebugremote32` (UNIX) (or `cobdebugremote64` when debugging 64-bit processes) in Visual COBOL.

To debug locally-developed programs on a remote machine you must start `cobdebugremote` (Windows) or `cobdebugremote32` (UNIX) (or `cobdebugremote64` when debugging 64-bit processes) before communication can be established. See the Visual COBOL help for more information on cobdebugremote.

To debug remotely-developed programs on a remote machine:

- If you have developed a COBOL remote project on a machine using a local Eclipse installation and a remote UNIX machine running Micro Focus Visual COBOL Development Hub, you use debug configurations in the same way as locally-developed projects. However you do not need to start the `cobdebugremote` (Windows) or `cobdebugremote32` (UNIX) process (or `cobdebugremote64` for a 64-bit process) or identify a port for it to listen on, merely select the remote project in the debug configuration.
- To enable remote debugging using a COBOL Application debug configuration, you need to have an X window implementation installed and running, so that the output of a remote application running on a UNIX machine can be viewed on your local machine. You do not need X window for other debug configuration types.

Source Pool View

The source pool view in Net Express showed all source files available in the project directory, regardless of whether or not they are used in the current build type. This view is not available in Visual COBOL.

Tips: Eclipse IDE Equivalents to IDE Features in Net Express

The following table shows Net Express IDE features and their corresponding equivalents and locations in Eclipse.

Functionality	In Net Express	In Visual COBOL for Eclipse
Project Control		
Project filename	`*.APP`	`.cobolProj` in Project directory
Add file to project		All files in the project directory and its subdirectories are automatically part of the project. You can exclude programs or subdirectories from a build using the **Build Action** option on the context menu.
Copybook path		By default, all directories within a project and their subdirectories are included in the initial copybook path.To make changes, click **Project > Project Properties > Micro Focus**

Functionality	In Net Express	In Visual COBOL for Eclipse
		COBOL > Build Path and select the **Copybook Paths** tab.
Build settings for the project: • COBOL • Preprocessor • Additional Directive		Right-click the project name and select **Properties**, (or click **Project > Properties** from the main menu).
Execution environment settings: • General • COBOL		The execution environment is COBOL Server.
Debug settings: • DateWarp • Stored Procedures		DateWarp is present in the debug launch configurations.
Editing		
Suggest Word/Content Assist	**CTRL+G**	**CTRL+Space**
Locate	**F12** (or context menu **Locate**)	**F3** (or context menu **Goto Definition**)
COBOL Find	**CTRL+Shift+F12** (or context menu **COBOL Find**)	**CTRL+Shift+G** (or context menu **Find References**)
Compress	Tool bar compress  (or context menu **Compress**)	Double-click on an item to jump to the line in the program.
Bookmark	**CTRL+F2**	Click **Edit > Add Bookmark**.
Compiling		
Single file Compile	**CTRL+F7** (or click **check mark** )	Right-click a program name and select **Single File Compile**.  **Note:** This applies to native code only.
Build	**F7** (or click **build** )	By default, Eclipse automatically builds projects to keep them up to date. You can also build on request. To turn off the automatic build, click **Project > Build Automatically**.
Build All	**ALT+B A**	Cleaning a project forces a rebuild. To access this, click **Project > Clean**.
Debugging		
Start Debugging	**Alt+D A**	Choose **Debug** or **F11** Before launching the session, create an appropriate debug launch configuration.
Stop Debugging	**Shift+F5**	**CTRL+F2** (or context menu **Terminate**)
Restart Debugging	**Ctrl+Shift+F5**	From the executable in the debug stack view, right-click and select **Relauch** from the context menu.
Run	**F5**	**CTRL+F11**

Functionality	In Net Express	In Visual COBOL for Eclipse
Step	**F11** (or click **step** )	**F5**
Step All	**Ctrl+F5**	**Resume**  on the debug toolbar.
Run Thru		**F6** (or **Step Over** )
Run Return		**F7** (or **Step Return** )
Run to Cursor	**Shift+F10** (or context menu)	**CTRL+R** (or context menu **Run to Line** )
Skip to Cursor	**CTRL+Shift+F10**	**ALT+F12, R** (or **Reset Execution Point** )
Skip Statement		No equivalent. Use **ALT+F12, R** (or **Reset Execution Point** )
Skip Return		No equivalent. Use **ALT+F12, R** (or **Reset Execution Point** )
Examine ' *data item'*	**Shift+F9**	The Variables View automatically shows variable values on the current and previous lines.
Breakpoint set	**F9**	Double click in the left margin to toggle a breakpoint.
Conditional Breakpoint		From the Breakpoints View, select a breakpoint and edit the properties.
Break on Data Change	Via list view	To set a **WatchPoint**, double-click in the margin of line containing a WS item, or set it from the Outline View context menu.

Upgrading from RM/COBOL®

There are a number of settings in Visual COBOL that are designed specifically to ensure that your existing RM/COBOL source code can compile and run in Visual COBOL.

The Visual COBOL help includes information about how to enable compatibility with RM/COBOL in Visual COBOL and convert your existing RM/COBOL applications. For more information, read *Compatibility with RM/COBOL* in the *Programming* section of the help.

In addition, please note the following:

- If you compile an RM/COBOL application in Visual COBOL with the RM dialect, you must explicitly use the standard COBOL calling convention to any programs that are not compiled for RM compatibility. This affects calls to CBL_ or CGIX routines, but the RM C$ routines already use the RM calling convention.

Appendix

Native COBOL Compared with Managed COBOL

Native COBOL and managed COBOL differ in how they compile and how the run-time management services, such as security, threading and memory management are provided.

Managed COBOL for JVM compiles to Java bytecode, and native COBOL compiles to machine code. This means that native COBOL has to be compiled for the operating system on which it is going to run, whereas compiled managed COBOL can run on any platform.

For JVM managed code, the management services are provided by JVM. For native COBOL, the management services are available in the operating system, and your code has to call the appropriate services depending on the operating system. The management services are:

- The Java Runtime Environment (JRE).
- The Java API, which provides a large library of building blocks to simplify many application programming tasks. It provides APIs for everything from dates and times to thread management. These features are less useful for procedural COBOL programs, but once you have migrated them to managed code, you can use them to add new features.
- Seamless interoperation of COBOL programs with programs in other managed languages.
- The ability to write object-oriented COBOL.

Building Native and Managed COBOL Applications

You use the IDE to develop, compile and debug your applications, for both native and managed code. You can write new COBOL code or you can recompile existing COBOL as 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. JVM COBOL applications are deployed to the platform running the JVM.

Customer Feedback

We welcome your feedback regarding Micro Focus documentation.

Submit feedback regarding this Help

Click the above link to email your comments to Micro Focus.