COBOL-IT Compiler Suite Release Notes

Please note: Upgrading to a major release (1.0, 2.0, 3.0, etc…) requires recompilation of all programs. Release notes for minor releases (2.x, 3.x, etc…) document recently added features in maintenancereleases (2.x.x, 3.x.x, etc…). In exceptional cases, a minor release may contain updates that require recompilation of all programs. Where this is the case, an *Important note* will be highlighted in the release notes.

Release

3.11

Supported platforms have changed for the 3.11 release. Please see the COBOL-IT Port list.

3.10

Version 3.10 is a *minor release*. Enhancements to runtime execution performance cause changes to generated code. Users upgrading to version 3.10 from a previous release are advised to recompile and test their complete system prior to implementing in production.

3.10.47

3.10.47 CitSORT supports IFTHEN clause for INREC, OUTREC

The IFTHEN clause, and BUILD subclause are supported in the INREC and OUTREC statements.

As an example:
Consider a case where a line sequential file contains a header and trailer record that have different formats than the main body of the sequential file, for example:

```
hdr 14-jun-2018
04James Madison         180903041817030408Democratic-RepublicanGeorge Clinton
03Thomas Jefferson      180103041809030408Democratic-RepublicanAaron Burr
02John Adams            179703041801030404Federalist           Thomas Jefferson
01George Washington     178904301797030408Independent          John Adams
trl   4 records total
```

A sample CitSORT command, including the INREC/IFTHEN and OUTREC/IFTHEN clauses, with BUILD subclauses:

```
USE PRES2.TXT RECORD F 85 ORG LS
INREC IFTHEN=(WHEN=(1,3,CH,EQ,C'hdr'), BUILD=(C'00',1,25)),
IFTHEN=(WHEN=(1,3,CH,EQ,C'trl'), BUILD=(C'05',1,25)),
IFTHEN=(WHEN=(1,2,NU,GE,00), BUILD=(1,85))
SORT FIELDS=(1,2,NU,A)
GIVE PRES3.TXT RECORD F 85 ORG LS
OUTREC IFTHEN=(WHEN=(1,2,CH,EQ,C'00'),BUILD=(3,23)),
IFTHEN=(WHEN=(1,2,CH,EQ,C'05'), BUILD=(3,23))
```

The first INREC/IFTHEN statement creates a condition when character 1-3 are "hdr". In this case, the the characters "00" are inserted at the beginning of the line followed by the characters 1 thru 25 of the line.

The second INREC/IFTHEN statement creates a condition when character 1-3 are "trl". In this case, the characters "05" are inserted at the beginning of the line followed by the characters 1 thru 25 of the line.

The third INREC/IFTHEN statement creates a condition for the modified body of data. The header file has been marked as record "00". The trailer file as record "05". So the third INREC/IFTHEN condition applies to all lines, identifed with the first two bytes being numeric and GE 0. The lines are re-written as-is.

The file, originally in descending format, is SORTed in ascending format
on the numeric fields in bytes 1-2.

The OUTREC/IFTHEN statement strips out the values that have been inserted in character positions 1-2. In records where the first two bytes are '00' or '05', the record is re-built from position

3.10.47 -fready-trace compiler flag

The -fready-trace compiler flag enables paragraph tracing between READY TRACE and RESET TRACE procedural COBOL statements. In the interval between the READY TRACE and RESET TRACE statements, paragraph tracing output is written to the console in the format:

PROGRAM-ID: [program-id]: [paragraph name]

3.10.47 -fnumval-validate compiler flag

The -fnumval-validate compiler flag validates argument 1 of the NUMVAL function.

3.10.41 --xdd-prefix=<dir> compiler flag

The –xdd-prefix compiler flag causes the XDD file created with the -fgen-xdd compiler flag to be stored in the directory named in <dir>. As an example:

>cobc -fgen-xdd –xdd-prefix=/my/doc/xdd hello.cbl causes all xdds generated in the compilation to be stored in the /my/doc/xdd folder.

3.10.39 CitSORT supports ALTSEQ CODE with TRAN=ALTSEQ

ALTSEQ Statement

The ALTSEQ statement is used to change the alternate translation table (ALTSEQ table). Modifications are applied to the standard ASCII/EBCDIC translation table. When using the ALTSEQ statement, the TRAN=ALTSEQ clause must be used in context (any segment list that can include an EDIT or FORMAT clause may include a TRAN=ALTSEQ clause) to apply the modifications described by the ALTSEQ statement.

General Format

[ALTSEQ CODE=({old-char1, new-char1}...)]

Examples:

ALTSEQ Statement used to modify data in copy of existing file

```
>CITSORT FIELDS=COPY USE PRESIDENTS.DAT RECORD F 85 GIVE TESTOUT.DAT
ALTSEQ CODE=(0540,0E40)
OUTREC FIELDS=(1,4,5,TRAN=ALTSEQ)
```

ALTSEQ Statement used with an INREC

```
>CITSORT FIELDS=(1,80,CH,A),SKIPREC=1
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
INCLUDE COND=(1,80,CH,NE,C'          ')
ALTSEQ CODE=(7F40,6D60)
INREC FIELDS=(1,80,TRAN=ALTSEQ)
SUM  FIELDS=NONE
OUTREC FIELDS=(2,64,16X)
```

ALTSEQ Statement used with an OUTREC

```
> SORT FIELDS=COPY
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
ALTSEQ CODE=(0040,0140,0240,0340,0440,0540,0640,0740,0840,0940)
OUTREC FIELDS=(1,80,TRAN=ALTSEQ)
```

ALTSEQ Statement used with a single value

```
CITSORT FIELDS=(1,80,CH,A),SKIPREC=1
USE PRESIDENTS.DAT RECORD F85 GIVE TESTOUT.DAT
INCLUDE COND=(1,80,CH,NE,C'          ')
ALTSEQ CODE=(7F40)
INREC FIELDS=(1,80,TRAN=ALTSEQ)
SUM  FIELDS=NONE
OUTREC FIELDS=(2,64,16X)
```

3.10.39 CitSORT support SKIPREC

SKIPREC Clause

The SKIPREC Clause indicates the number of records to skip before beginning to process the SORT.

General Format

[SKIPREC=<number>]

Syntax

1. <number> is an integer describing the number of records to skip before beginning to process the SORT.

General Rules

1. When SKIPREC=<number> is indicated, the SORT begins process at record <number> +1.

Example

>CITSORT USE PRESIDENTS.DAT RECORD F 85 SORT FIELDS (1,2,A) FORMAT=NU SKIPREC=1 GIVE PRES2.TXT ORG LS

3.10.35 Using COB_DEBUG_STARTUP_FILE=<filename> and the replace command

The console debugger cobcdb can now locate source files that have been re-located after compilation using the COB_DEBUG_STARTUP_FILE runtime environment variable and invoking the replace debugger command.

The COB_DEBUG_STARTUP_FILE can be set to the name and location of a file containing any number of commands that are executed when cobcdb is started.

export COB_DEBUG_STARTUP_FILE=<filename>

To locate a source file that has been moved, and associate it with an object compiled for debug, use the 'replace' command, which changes the path to the source file.

The syntax is as follows: replace <oldprefix> : <newprefix>

The replace debugger command allows you to replace the location where the source files associated with the program being debugged are stored.

The replace debugger command replaces any prefix of the full pathname, so the command replace /dirA : /dirB will allow any program that was originally compiled in /dirA/dev/sources to have its source stored in /dirB/dev/sources.

Subsequent commands are stacked, so when typing two more commands as follows :

replace /dirC : /dirD

replace /dirE : /dirF

you will end up with a list of three possible replacements. Only the first matching replacement will be executed.

Further usages include:

replace <no arguments>	Resets the list, removing active replacements
replace ?	Produces a list of active replacements.

Note that replace only affects the output of the list command. Other commands such as info sources or break will still produce the original pathname as it was stored in the binary code of the program. Other commands such as break require a match with the original pathname in order to be executed.

3.10.35 -fxparse-event

The -fxparse-event compiler flag causes the XML PARSE statement to generate START-OF-DOCUMENT and END-OF-DOCUMENT XML-EVENTS.

3.10.35 Support for SUCCESS and FAILURE conditions

For compatibility with VAX/VMS COBOL, the internal conditions SUCCESS and FAILURE are now supported. As a result, SUCCESS and FAILURE are now reserved words.

The SUCCESS and FAILURE internal conditions are used as follows:

if <variable > is SUCCESS then...
if <variable > is FAILURE then...

To determine SUCCESS or FAILURE, the Least significant bit in the variable is tested.

1 is SUCCESS
0 is FAILURE

3.10.35 -dump-config

The -dump-config compiler flag has been updated so that the report includes all of the most recently added compiler configuration settings.

The -dump-config compiler flag dumps all of the compiler config file settings being used in the compilation session to stdout. The -dump-config compiler flag can be used with or without a COBOL source file.

To dump default compiler configuration file setting to a file:

>cobc –dump-config > [dumpfile].

To dump compiler configuration file settings for the current compilation:

>cobc [compiler flags] -dump-config hello.cbl > [dumpfile].

Note that the results of this update will also be seen in the listfile, where the compiler configuration file settings for the current compilation are listed.

3.10.29 Advice : Microsoft Visual C++ 20xx Redistributable Package removed

For reasons related to licensing, the Microsoft redistributable C runtime has been removed from the COBOL-IT package. Users that require the Microsoft redistributable Package must download it directly from the Microsoft website.

The Microsoft Visual C++ Redistributable Package installs the runtime components of Visual C++ libraries required to run applications developed with Visual C++ on a computer that does not have

the same version of Visual C++ installed. As an example, COBOL-IT applications that are developed using Microsoft Visual C++ 2010 require that the Microsoft Visual C++ 2010 Redistributable package be installed in order to run.

This package is available to the user if they have Microsoft Visual C++ installed on their computer.

If they do not, they must download the appropriate Microsoft Visual C++ Redistributable Package from the Microsoft website.

3.10.29 Number of Regions increased to 100

The upper limit on the number of regions that can be processed concurrently by CICS has been increased to 100. The upper limit was formerly 16 concurrent regions.

3.10.29 COB_RUNTIME_CHECK_TRACE=[Y/N/Module list separated by ; or : (Windows)]

The syntax for use with the COB_RUNTIME_CHECK_TRACE environment variable now includes the ability to associate COB_RUNTIME_CHECK_TRACE with a list of modules, separated by a semi-colon ";" or colon ":" (Windows). Note that the module list must be surrounded by single-quotes in Linux/UNIX, and not surrounded by single-quotes in Windows.

In Linux/UNIX, the module list must be surrounded by single-quotes:
Example:
>export COB_RUNTIME_CHECK_TRACE='module1;module2'

In Windows, the module list must not be surrounded by single-quotes, and both the ";" and ":" characters are accepted as module delimiters.
Example:
>SET COB_RUNTIME_CHECK_TRACE=module1;module2
or
>SET COB_RUNTIME_CHECK_TRACE=module1:module2

3.10.29 -fcall-lowercase

The -fcall-lowercase compiler flag affects the handling of literals that are the target of a CALL statement. As an example, consider the literal "MyProg" in the statement: CALL "MyProg". In this case, the -fcall-lowercase compiler flag causes all of the characters in the literal "MyProg" to be converted to lowercase.

3.10.29 call-lowercase: [yes/no]

Default is `call-lowercase:no`

The call-lowercase compiler configuration flag affects the handling of literals that are the target of a CALL statement. As an example, consider the literal "MyProg" in the statement: CALL "MyProg".

When set to yes, all of the characters in the literal "MyProg" are converted to lowercase.
When set to no (the default), this setting is ignored.

3.10.29 -fcall-uppercase

The -fcall-uppercase compiler flag affects the handling of literals that are the target of a CALL statement. As an example, consider the literal "MyProg" in the statement: CALL "MyProg". In this case, the -fcall-uppercase compiler flag causes all of the characters in the literal "MyProg" to be converted to uppercase.

3.10.29 call-uppercase: [yes/no]

Default is `call-uppercase:no`

The call-uppercase compiler configuration flag affects the handling of literals that are the target of a CALL statement. As an example, consider the literal "MyProg" in the statement: CALL "MyProg".

When set to yes, all of the characters in the literal "MyProg" are converted to uppercase.
When set to no (the default), this setting is ignored.

3.10.29 COB_FILE_CASE=[UPPER|LOWER]

The runtime environment variable COB_FILE_CASE=[UPPER|LOWER] forces all file names to be converted to upper/lower case.

3.10.29 COB_FILEMAP_CASE=[UPPER/LOWER]

The runtime environment variable COB_FILEMAP_CASE=[UPPER|LOWER] forces the runtime to convert all literals that are the target of ASSIGN EXTERNAL clauses that are associated with environment variables to upper/lower case before trying to resolve the name of the environment variable associated with the file.

Example:
SELECT myfile ASSIGN EXTERNAL "FileA"….

When there is no setting of COB_FILEMAP_CASE, the runtime looks for an environment variable named FileA.

When COB_FILEMAP_CASE=UPPER, the runtime looks for an environment variable named FILEA.

When COB_FILEMAP_CASE=LOWER, the runtime looks for an environment variable named filea.

3.10.29 -fvalue-of-id-priority

The -fvalue-of-id-priority compiler flag gives priority to the literal or data element named in the VALUE OF FILE-ID clause in the FD, causing the target of the VALUE OF FILE-ID clause in the FD to override the target of the ASSIGN clause for the file.

3.10.29 value-of-id-priority: [yes/no]

Default is `value-of-id-priority:no`

The value-of-id-priority compiler configuration flag gives priority to the literal or data element named in the VALUE OF FILE-ID clause in the FD.

When set to yes, the literal or data element that is the target of the VALUE OF FILE-ID clause in the FD overrides target of the ASSIGN clause for the file.
When set to no (the default), this setting is ignored.

3.10.26 FUNCTION REVERSE works with PIC N

FUNCTION REVERSE has been enhanced so that it supports USAGE NATIONAL arguments. The intrinsic FUNCTION REVERSE reverses the order of the characters of the argument.

3.10.26 XDD generation supports multi-part keys

The XDDs created with the -fgen-xdd compiler flag, and used by the Data Displayer now support multi-part keys.

3.10.17 Variable Value displays of table element when subscript is local variable

When the index of a table element is expressed as a local variable, the debugger will alter the value display depending on the value of the index.

As an example:
01 Idx PIC 999.
01 Array PIC X(10) OCCURS 1 TO 100.

Array(Idx) can be Watched in the Expression View, and the value of Array(Idx) will change depending on the value of Idx.

3.10.15 -fcompute-ibm / -fno-compute-ibm-trunc conformance enhanced

Conformance to the COMPUTE rules defined by IBM has been expanded to include cases where COMPUTE statements involve a data item described as USAGE DISPLAY (no decimals) and a data item described as USAGE COMP-2.

3.10.14 -fcompute-ibm / -fno-compute-ibm-trunc

The -fcompute-ibm compiler flag has increased conformance to the COMPUTE rules defined by IBM which are applied when the -fcompute-ibm compiler is used. Specifically, every intermediate result is truncated respecting mainframe rules. This can be disabled by using the -fno-compute-ibm-trunc compiler flag.

3.10.14 Disam extfh driver compatible with MF C-ISAM files

Programs compiled with the -fdisam compiler flag can be used to read and write C-ISAM indexed files created by Micro Focus applications. Such files do not need to undergo any conversions.

3.10.14 DISAM files created without .dat extension by default

When compiling with -fdisam, and creating new DISAM indexed files, the default behavior of an OPEN OUTPUT statement is for the DISAM file to be created without a .dat extension on the data file of the indexed file. This behavior is compatible with Micro Focus default behaviors. The behavior of an OPEN INPUT/I-O statement is to first check for a file with no .dat extension and then, if none is found, to check for a file with .dat extension. You can force the DISAM indexed files to be created with a .dat extension on the data file by setting the runtime environment variable COB_DOT_DAT=y.

3.10.14 COB_DOT_DAT [y/n]

The COB_DOT_DAT runtime environment variable, when set to y, forces a .dat extension to be appended to a DISAM indexed file when it is created. When set to n (the default), no .dat extension is appended to the DISAM file when it is created.

3.10.13 -temps-dir <path>

The -temps-dir <path> compiler flag lets you specify where to store temporary files that you do not want to be erased. This includes files that would be stored in the c subdirectory by default, when compiling with -save-temps, for example.

3.10.13 XML PARSE supports ENCODING "code page"

The XML PARSE statement supports the parsing of XML documents that contain XML encoding declarations that specify code pages as defined by IBM.

3.10.11 COB_RUNTIME_CHECK_TRACE [Y/N]

This environment variable should be used with caution.

The COB_RUNTIME_CHECK_TRACE runtime environment variable affects the behavior of the "subscript out of bounds" runtime check that is made when compiling with -g -debug, and optionally with -fmem-info.

When set to "Y", the "subscript out of bounds" will not force the runtime to abort, but to generate a message referencing the line on which the condition was detected. Messages generated will be written to the COB_ERROR_FILE if it is defined, otherwise, they will be written to stdout. No memory dump is produced, even if the COB_DUMP environment variable is set and the program is compiled with -fmem-info.

Consider a case where a program has a table with 5 elements, but a PERFORM loop increments the subscript 10 times. When COB_RUNTIME_CHECK_TRACE is set to N (the default), the "subscript out of bounds" condition forces the runtime to abort when the first out of bounds condition is detected. If the program is compiled with -fmem-info and the COB_DUMP environment variable is set, a memory dump is written to the COB_DUMP file.

When COB_RUNTIIME_CHECK_TRACE is set to Y, the runtime will continue to run, producing output as the subscript continues to increment, as seen below:

```
C:/COBOL/COBOLIT/Samples/test.cbl:20: libcob: Subscript of 'sample-element' out of bounds: 6
C:/COBOL/COBOLIT/Samples/test.cbl:20: libcob: Subscript of 'sample-element' out of bounds: 7
C:/COBOL/COBOLIT/Samples/test.cbl:20: libcob: Subscript of 'sample-element' out of bounds: 8
C:/COBOL/COBOLIT/Samples/test.cbl:20: libcob: Subscript of 'sample-element' out of bounds: 9
C:/COBOL/COBOLIT/Samples/test.cbl:20: libcob: Subscript of 'sample-element' out of bounds: 10
```

3.10.11 CITSORT FILLBYTE

Requires CITSORT version 2.45

When FILLBYTE is specified, the output record will be padded (if needed) with that byte value instead of space.

Examples :

GIVE Statement, Reformatting an output record, with FILLBYTE

This example shows how the GIVE statement can be used with FILLBYTE. In the example below, trailing spaces will be filled with the "*" character (X'42').

```
> CITSORT COPY USE PRESIDENTS2.TXT ORG LS RECORD V 12,24 GIVE PRESIDENTS3.TXT
ORG SQ RECORD F 85 FILLBYTE 42
```

OUTREC Statement, Reformatting an output record, with FILLBYTE
This example shows how an OUTREC statement can be used with the FILLBYTE clause. In the example below, trailing spaces will be filled with the "*" character (X'42').

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=('PRESIDENT # ',1,2,' ',3,21), FILLBYTE 42
```

3.10.11 Visual C 2008 no longer supported in Windows.
Visual C 2008 is no longer supported in Windows.

3.10.11 -fkeep-unused compiler flag
The -fkeep-unused compiler flag causes memory to be allocated for the field tree of level-01 and level-77 data items that are declared which contain sub-fields and in which none of these sub-fields is used.

The -fno-keep-unused compiler flag causes memory to not be allocated for the field tree of level-01 and level-77 data items that are declared which contain sub-fields and in which none of these sub-fields is used.

3.10.11 keep_unused: [yes/no]
Default is `keep-unused: yes`

The keep-unused compiler configuration flag affects how memory is allocated when the runtime is initiated.

When set to yes (the default), memory is allocated for field tree of level-01 and level-77 data items that are declared which contain elementary sub-fields and in which none of these sub-fields is used.

When set to no, memory is not allocated for the field tree of level-01 or level-77 data items that are declared which contain elementary sub-fields in which none of these sub-fields is used.

3.10.11 -fobj-cit behaviors
In a runtime session in which some of the programs have been compiled with the -fobj-cit compiler flag, the runtime, when resolving the names of programs/sub-programs, will search first for programs/sub-programs with the .cit extension. Subsequently, the runtime will search for programs/sub-programs with the .so/.dll extension.

In a runtime session in which no programs have been compiled with the -fobj-cit compiler flag, the runtime will search first for programs/sub-programs with the .so/.dll extension.

3.10.11 -floosy-comment
The -floosy-comment compiler flag allows a * in column 8 to be used to mark a comment.

3.10.11 non-ibm-5.2-syntax: [yes/no]

Default is `non-ibm-5.2-syntax: no`

The non-ibm-5.2-syntax compiler configuration flag affect the syntax that is recognized by the compiler.

When set to yes, the non-ibm-5.2-syntax compiler configuration flag restricts syntax to IBM ENTERPRISE 5.2 COBOL.

3.10.8 Sun Studio replaced by gcc provided with Sun Solaris

Sun Studio is no longer supported with Sun Solaris 10 Sparc ports. It has been replaced by the gcc provided with the Sun Solaris distribution.

3.10.8 D-ISAM Support

Note- The VBISAM engine will be deprecated in the next major release (4.x) and replaced by the D-ISAM indexed file engine. The D-ISAM engine is more stable and more widely used than the VBISAM engine, and is fully compatible with IBM C-ISAM 7.2. Unfortunately, VBISAM files are not readable by D-ISAM and will require conversion. See details below for setting the COB_EXTFH environment variable at runtime to direct the runtime whether to use the VBISAM Extfh driver (on READs) or the D-ISAM Extfh driver (on WRITEs) in your conversion.

3.10.8 Auto detection and load of EXTFH library

At run time, if the COB_EXTFH environment variable is defined (and no additional variables are included to specify what library to load), the runtime looks for lib$(COB_EXTFH) on UNIX and $(COB_EXTFH)_dll.dll on Windows in the COBOL-IT installation directory and all directories indicated in the COB_LIBRARY_PATH.

As an example, to use D-ISAM through EXTFH :
export COB_EXTFH=disamextfh
Libdisamextfh.so will be found and loaded into the CIT distribution directory (COBOLITDIR).

The current CIT distribution includes the following EXTFH drivers:
BerkelyDB based : bdbextfh
VBIsam based : vbisamextfh (provided for compatibility, will be deprecated in the next major release (4.x))
D-ISAM based : disamextfh

These file systems can be used with all COBOL-IT tools including CitSORT.

3.10.8 -fdisam

The -fdisam compiler flag forces use of the D-ISAM Extfh indexed file engine. In the next major release (4.x), the -fdisam compiler flag will be set by default.

3.10.8 disam: [yes/no]

Default is `disam: no`.
The disam compiler configuration flag when set to yes, forces use of the D-ISAM Extfh indexed file engine. In the next major release (4.x), the default will be set to yes.

3.10.8 dcheck

The D-ISAM indexed file engine includes a check utility: dcheck. Dcheck is available on all Linux, UNIX and Windows systems.

3.10.8 cob_profiling_<PID>_final.xls

By default, the .xls file output by the Profiler is no longer generated for each module at the exit of the module. Instead, a single .xls file is output at the program exit.
The single output file is created using the naming convention:
cob_profiling_<PID>_final.xls.

This change improves performance significantly by reducing the profiler overhead in cases where subprograms are called.

3.10.8 COB_PROFILING_EACH_MODULE

The COB_PROFILING_EACH_MODULE runtime environment variable, when set to Y, causes the profiler to revert to the old profiling behavior, in which the .xls file is output at the exit of each module in an application.

3.10.8 *PRAGMA "DUMP" "a label"

Adding *PRAGMA "DUMP" "a label" to your source code forces a a profiling dump with file name cob_profiling_<PID><a label>.xls each time the program runs that line.

3.10.8 Dynamic Profiling

A program compiled using the -fprofiling compiler flag may be attached exactly like a program to debug (with cobcdb -p <pid>) even if the program is not compiled for debugging .

3.10.8 Info profiling

The cobcdb compiler command `>info profiling` causes a profiling dump to be produced, dumping profiling information at the current point in the program. Profiling information is dumped in the .xls file format.

3.10.5 -ffdclear

The -ffdclear compiler flag causes the record to be INITIALIZEd after each WRITE.

3.10.5 -fline-seq-dos

The -fline-seq-dos compiler flag affects the writing of the record delimiter at the end of each record in a line sequential file. When compiling with -fline-seq-dos, the record delimiter is set to <CR><LF>.

3.10.5 Code generation optimizations

Code generation has been updated to improve performance.
Entries below show how to enable the optimizations by using the -O compiler flag. The -O2 compiler flag can also be used to enable optimizations at the "C" compiler level.

3.10.5 -O, -O2 compiler flags

The -O and -O2 compiler flags now automatically enable the following compiler flags by default: -fbin-opt, -fcmp-opt, -fcmp-inline, -fp-opt, -finitialize-opt, -foptimize-move, -foptimize-move-

call, -ffast-op, -findex-optimize, -fdecimal-optimize. For more details about each of these optimizing compiler flags, see below.

3.10.5 -fbin-opt

Using the -fbin-opt compiler flag enables the use of CPU integers when manipulating USAGE COMP and USAGE COMP-5 data elements.

3.10.5 -fcmp-opt

Using the -fcmp-opt compiler flag causes comparisons to apply faster algorithms when applicable. (Default is on)

3.10.5 -fcmp-inline

Using the -fmp-inline compiler flag causes comparisons to be inined in the C code instead of through calls to the runtime library when possible.

3.10.5 -fp-opt

Using the -fp-opt compiler flag causes COMP-2 operations to be inlined in C, and maximizes the use of the CPU Floating Point unit.

3.10.5 -finitialize-opt

Using the -finitialize-opt compiler flag optimizes the implementation of the initia field initialization at runtime startup and the execution of the INITIALIZE statement by grouping field initializations wherever possible.

3.10.5 -foptimize-move

Using the -foptimize-move compiler flag causes MOVE operations to be optimized by -fmem-info where the source and target fields have identical declarations. (Default is on).

3.10.5 -foptimize-move-call

Using the -foptimize-move-call compiler flag causes MOVE operations to be optimized by pre-selecting the internal runtime library routines used for the MOVE when possible.

3.10.5 -ffast-op

Using the -ffast-op compiler flag enables the runtime to use faster operations when manipulating data items declared as USAGE DISPLAY or USAGE COMP-3.

3.10.5 -findex-optimize

Using the -findex-optimize compiler flag improves performance where indexes in tables are evaluated and USAGE DISPLAY fields are used as indexes. In these cases, the index values are cached in a C integer field to improve performance.

3.10.5 -fdecimal-optimize

Using the -fdecimal-optimize compiler flag enables the optimization of decimal operations in the COMPUTE statement.

3.10.5 Comments beginning with *> accept both ‘ and “ quote notations

When using the *> notation to designate a comment, you may now use both single-quote, and double-quote quote notations in your comment.

3.10.5 Locating the <debugdb> file

The runtime checks for the <debugdb> file using the name given at compilation time. If not found, the runtime will then check for the filename in the COB_LIBRARY_PATH location.

3.10.5 COB_NO_DOT_DAT environment variable

The COB_NO_DOT_DAT runtime environment variable, when set to Y, instructs the VBISAM indexed file driver to not append .dat to the data file. The index file still is created with a .idx extension.

3.10.5 COB_PROFILING_DIR environment variable

When compiling with the -fprofiling compiler flag, the runtime will check for the COB_PROFILING_DIR environment variable, and generate the profiling data file in that directory if it is defined. Otherwise, the profiling data file is created in the same directory as the source file.

Note that COB_PROFILING_DIR environment variable requires a trailing slash.
As an example, in Linux/UNIX environments:
>export COB_PROFILING_DIR=mydir/

As a result, the output file is generated as mydir/cob_profiling_<pid>_final.xls

In Windows environments:
>set COB_PROFILING_DIR=mydir\

As a result, the output file is generated as mydir\cob_profiling_<pid>_final.xls

3.10.5 -frw-mode-nopf compiler flag

The -frw-mode-nopf compiler fag is equivalent to setting MODE NOPF for a Report Section report. MODE NOPF causes the Report Writer to emulate an external print driver that does not remove printer control characters, like FF. When using the -frw-mode-nopf compiler flag, the report file is written as a standard LINE SEQUENTIAL file.

3.10.5 -frw-mode-nopf-dos compiler flag

The -frw-mode-nopf-dos compiler flag is equivalent to setting MODE NOPF for a Report Section report. MODE NOPF causes the Report Writer to emulate an external print driver that does not generate printer control characters, like FF. When using the -frw-mode-nopf-dos compiler flag, the report file is generated with fixed-length lines that are padded with SPACES and use the CR/LF record delimiter.

Fixes

3.10.47 SEARCH ALL searching empty table

SEARCH ALL would launch a SEARCH on an empty table.

This has been corrected. If a table is empty, the SEARCH ALL statement will not start the process of searching.

3.10.47 Division operator could produce incorrect results not using -fcompute-ibm

When the -fcompute-ibm compiler flag was not used , the division operator could produce incorrect results in some situations.

This has been corrected.

3.10.47 COPY REPLACING LEADING/TRAILING with more than one word

COPY REPLACING LEADING/TRAILING with more than one word is now supported.

As an example:

```
COPY "wscopy.cpy"
     REPLACING LEADING ==ws== by ==xx==
               LEADING ==pr== by ==yy==.
```

3.10.46 Memory dump does not repeat OCCURS element of same value

ID: 1149726596

In a memory dump OCCURS elements in a table that possessed the same value were not repeated, and so were not accessible.

Example :
WORKING-STORAGE.Data(01) = <Low-Value>
WORKING-STORAGE.Data(02) to (10) = <Same as above>

This has been corrected. OCCURS elements in a table in the debugger now evaluated and displayed.

Before this fix, if you had:
01 idx PIC 99.
03 tab1 OCCURS 10
05 Data

You had to evaluate `tab1(ixd).Data` to see the correct value. Now you may evaluate `tab1.Data(ixd)`. This is a more common COBOL formulation. In the COBOL-IT Developer Studio Debugger perspective, this latter formulation is now understood, as the debugger will associate the idx to tab1.

3.10.46 XML GENERATE statement with ampersand sign produces wrong output

ID: 1149818236

If the Ampersand character was in the in put string, the output of COBOL-IT's XML GENERATE statement did not surround the & character with quotes, "&". As a result this output was not portable to the IBM implementation of XML GENERATE.

This has been corrected.

3.10.46 Impact of source-location: yes configuration flag for the debugger

ID: 1149887600

When using the source-location: yes compiler configuration flag, programs compiled without -g were visible in the debugger.

This has been corrected. When using source-location:yes without the use of the -g compiler flag programs are not in the debugger, but the name of the program and line number are dumped when an error occurs.

3.10.45 Function IS NUMERIC is incorrect in combination with TRIM functions*

ID: 1149896512

The IS NUMERIC check would incorrectly identify an empty string as being NUMERIC.

This has been corrected. The IF IS NUMERIC test now returns FALSE if the field size is null.

3.10.45 Form feed incorrectly written using USAGE NATIONAL & UTF16BE

ID: 1149818932

The output from a WRITE AFTER ADVANCING statement- when combining USAGE NATIONAL and the UTF16BE compiler configuration flag could cause the form feed to be represented in a single byte instead of in two bytes.

This has been corrected.

3.10.45 Internal Error: can't convert from utf-8 1.0 : U_ILLEGAL_ARGUMENT_ERROR

ID: 1149877646

The U_ILLEGAL_ARGUMENT error could be returned by the runtime when executing an XML PARSE statement in combination with a file containing a table with an OCCURS DEPENDING ON clause.

This has been corrected.

3.10.44 File cannot have both EXTERNAL and GLOBAL clauses

ID: 1149874078

In a case where a program contained both “IS EXTERNAL” and “IS GLOBAL” file descriptions, e.g.
fd myfile is global is external.

The compiler would return an error:
Error: File cannot have both EXTERNAL and GLOBAL clauses

This has been corrected. IS EXTERNAL and IS GLOBAL can be applied to the same file.

3.10.44 Access to Level-88 declaration in TYPEDEF not possible

ID: 1149874134

When a level-88 is created under a sub-element in a TYPEDEF, the level-88 was not recognized. As an example:

```
01  TYPEDEF-02          IS TYPEDEF.
  05 LVL2  PIC 9(1).
```

```
88  NUM-VAR01          VALUE ZERO.
88  NUM-VAR02          VALUE 1.
```

This has been corrected. Level 88's are now propagated inside TYPEDEFs.

3.10.43 -foptimize-move-call compiler flag causes error in online program

ID: 1149858048

In some situations, the -foptimize-move-call compiler flag could produce an error in an on-line program with Tuxedo, as it could cause the commarea returning to the caller to be corrupted. In this situation, excluding the -foptimize-move-call compiler flag would resolve the problem, but there would be a performance penalty.
This has been corrected.

3.10.37 SIZE IS statement produces compile error in Windows 10

ID: 1149651916

In Windows 10, the SIZE IS clause would produce a compile error "Unexpected literal, expecting identifier.

This has been corrected.

3.10.37 OPEN file with environment variable in the name

ID: 1149522938

When defining a file name that includes the format "directory_name:filename", where "directory_name" is taken from an environment variable, the runtime now inserts a "/" between directory_name and filename when resolving the filename.

This has been corrected.

3.10.37 non-ibm-5.2-syntax: [error/warning/ok]

ID: 1149704244

Corrections were made to the template used for detecting non-ibm-5.2 syntax by the compiler.

3.10.37 mf-optional-file doesn't work with disam files

ID: 1149722884

The mf-optional-file:yes setting causes the OPEN EXTEND of a non-existent file to OPEN the file, while returning an error 05. Whe using a disam file, the OPEN failed with an error 35.

This has been corrected.

3.10.37 Lower case file extensions with COB_FILEMAP_CASE=upper

ID: 1149662172

When using the runtime environment variable COB_FILEMAP_CASE=upper with index files, files were created with the suffix ".dat" and ".idx" instead of ".DAT" and ".IDX" as expected.

This has been corrected

3.10.37 citsort take command produces memory dump on Sun SPARC

ID: 1149574980

>citsort take sortcommand.txt 2>&1 1>sortsyout.txt command could produce a cobol memory dump on a Sun SPARC system. The problem did not reproduce on Windows or Linux systems.

This has been corrected.

3.10.33 Length of 'XML-TEXT' out of bounds

Using LENGTH OF XML-TEXT in Reference Modification notation could produce unexpected results, including reference out-of-bounds error, when compiling with -debug.
As an example:
MOVE XML-TEXT(1:LENGTH OF XML-TEXT) TO VARIABLE-1.

This has been corrected.

3.10.32 XML-EVENT/XML-TEXT do not update correctly in Expression View

In the Debugger Studio, when you put a WATCH on XML-EVENT and XML-TEXT in order to monitor changing values in the Expression View, the fields in the Expression View would not update when values changed.

This has been corrected.

3.10.31 LENGTH OF XML-TEXT returns always 1

When using XML PARSE to get a value for an internal field XML-TEXT, the LENGTH OF statement did not return the correct length of the string. Instead, it always returned the value 1.

This has been corrected.

3.10.30 Runtime Crash after DISPLAY of PIC X(32767) variable

In some siituations the runtime would abort when DISPLAYing a variable described as PIC X(32767).

This has been corrected.

3.10.29 Error when displaying to UTF-8 terminal

An error was detected when displaying to a UTF-8 terminal.

This has been corrected.

3.10.29 source-codepage compiler flag accepts utf8 and utf-8 as synonyms

The -source-codepage compiler flag now considers utf8 and utf-8 to be synonyms and accepts both.

3.10.29 Some issues when using debugger on Windows

A number of minor issues were reported using the debugger on Windows.

These have been corrected.

3.10.29 -debug bounds check could return false positive

In some cases, the bounds checking performed when using the -debug compiler flag could incorrectly report an out-of-bounds condition on an INITIALIZE statement.

This has been corrected.

3.10.29 Crash when calling CBL_ALLOC_MEM

It was possible for a CALL “CBL_ALLOC_MEM” to cause the runtime to crash on a SPARC system if the USAGE POINTER data element passed to CBL_ALLOC_MEM was not aligned in memory.

This has been corrected.

3.10.29 cobcdb –listdid closes the running process on Windows

In some situations, running the command >cobcdb -listdid in Windows could close the running process.

This has been corrected.

3.10.29 OPEN INPUT SHARING WITH NO OTHER fails to open the file

The OPEN INPUT [filename] SHARING WITH NO OTHER syntax could fail to open the file.

This has been corrected.

3.10.29 Data allocation error in XML_PARSE

A data allocation using XML_PARSE was incorrect.

This has been corrected.

3.10.29 Parsing of C:\file… on windows

C:\file was incorrectly interpreted as $C\file where $C was replaced by the content of an environment variable C.

This has been corrected.

3.10.26 Minor problems corrected in the Windows debugger

Some minor problems were corrected in the WINDOWS debugger

3.10.19 WRITE AFTER/BEFORE anomaly when using ASSIGN TO PRINTER

When using an ASSIGN TO PRINTER clause, it was possible to get unexpected results when a WRITE BEFORE 1 LINES statement was followed by a WRITE AFTER 2 LINES statement. Then, in some cases, when using version 3.10.x, a line feed could be left out.

This has been corrected.

3.10.19 TYPEDEF fails to duplicate LEVEL 88

When a data description that has an IS TYPEDEF declaration, and contained a LEVEL-88 data name, the associated data name using the USAGE [typedef] clause did not preserve the LEVEL-88 data name.

This has been corrected.

3.10.19 XML GENERATE generation error

The rules governing the generation of a numeric field were incompatible with IBM generation rules in the case where a numeric field was declared with a PIC clause that was greater than needed for the given value. That is:

FieldA PIC 9(3) VALUE 1
Was being generated as : <FieldA>001</FieldA>

Whereas IBM COBOL would cause the following to be generated : <FieldA>1</FieldA>

This has been corrected.

3.10.17 Caching issues when evaluating expressions in the Debugger

It was possible for incorrect values to be displayed when evaluating expressions in the debugger due to caching issues in some circumstances.

This has been corrected.

3.10.17 Data Displayer would not display FD with only one field

The Data Displayer would not display data in cases where an FD consisted of a single field, such as:
FD test-file.
01 test-record pic x(80).

This has been corrected.

3.10.17 Incorrect Mapping of CodePage

When using the -futf-8 compiler flag with the -codepage “xxx” compiler flag, the CodePage was not correctly mapped.

This has been corrected.

3.10.17 Improved Compatibility with Mainframe behavior using -fcompute-ibm

When compiling with -fcompute-ibm, compatibility with Mainframe behavior was not correct in some cases where there was a combination of truncation in intermediate results and rounding.

This has been corrected.

3.10.16 CITSORT regression parsing Record Clause

In certain situations, CitSORT could terminate with an error clause referencing the RECORD clause inappropriately.

This has been corrected.

3.10.16 -fcompute-ibm intermediate result truncated incorrectly

When compiling with -fcompute-ibm, an intermediate result in a COMPUTE statement was detected that was truncated incorrectly.

This has been corrected.

3.10.16 SPARC CPU exception when compiling with -O

It was possible, in rare cases, for an object file compiled with -O to cause a CPU exception to be raised on some SPARC machines

This has been corrected.

3.10.15 DISAM driver would add .dat as default extension to data file

In some cases, the DISAM EXTFH driver would add .dat as a default extension to a data file.

This has been corrected. The default behavior for the DISAM driver is to not add .dat as a default extension to a data file.

3.10.15 Normal exit when debugging a program returns non-zero status

In some cases, when debugging a program, a normal exit could return a non-zero status.

This has been corrected.

3.10.15 Multiple calls to cobtidy cause runtime to crash

In some cases, multiple calls to the runtime API cobtidy function could result in a runtime crash.

This has been corrected.

3.10.15 Regression error in CITSORT 1.2.46

A regression error was detected in CitSORT 1.2.46 provided with version 3.10.14.

This has been corrected.

3.10.15 mf.conf correction

When using the -std=mf compiler flag, optional-file was incorrectly set to yes.

This has been corrected. The mf.conf file now correctly sets optional-file:no

3.10.15 Internal SORT using INPUT PROCEDURE error

A regression error was detected in the Compiler 3.10.14 in the internal sort when INPUT PROCUDURE was used.

This has been corrected.

3.10.14 Updates to dd_extfh

Updates to the dd_extfh EXTFH server used by the Data Displayer have been made to support new functionalities and to correct some issues.

3.10.13 Comparison of COMP-2 fields

The comparison 2 COMP-2 fields when no optimization was active could return the wrong results after version 3.10.5. As a result, after version 3.10.5, when comparing 2 COMP-2 fields, it was recommended that you compile with -O or -ffp-opt.

The problem has been corrected, and compiling with -O or -ffp-opt is no longer required when comparing 2 COMP-2 fields.

3.10.13 Truncation on USAGE COMP fields

In some situations data fields declared as USAGE COMP that contained more that 9 digits could be truncated.

This has been corrected.

3.10.13 SET FIELD TO ENTRY "Mylib.DLL" anomaly]

When a DLL has no entry point "Mylib", the SET FIELD TO ENTRY "Mylib.DLL" statement would still cause the dll to be loaded in memory, and another entry in the DLL might be used by the CALL statement.

This has been corrected.

3.10.13 Windows CALL to a __stdcall function accepts "@" character

In Windows, a CALL to a___stdcall function that contained an "@" character would not be resolved correctly.

This has been corrected. As an example, the following call is resolved correctly now:

CALL 66 "_MyFunc@4" USING Field

3.10.11 Consecutive calls to cob_tiny_rtd causes runtime to abort

In some cases, consecutive calls to the cob_tiny_rtd runtime API function could cause the runtime to abort.

This has been corrected.

3.10.11 Unstring anomaly with a source NATIONAL (Pic N)

In some cases, the results of an UNSTRING perform on as string stored as USAGE NATIONAL (PIC N) could be incorrect.

This has been corrected.

3.10.11 -fcopy-exec-replace could disable debugging

When compiling with -fcopy-exec-replace, in some cases, the debugger could become disabled.

This has been corrected.

3.10.11 -fnull-param anomaly when setting return size

When compiling with -fnull-param, in some cases, setting return size did not produce the expected result.

This has been corrected.

3.10.11 ACCEPT anomaly when using DECIMAL-POINT IS COMMA

It was possible for an ACCEPT statement to truncate input in programs using DECIMAL-POINT IS COMMA in which the data input to the ACCEPT statement is numeric and contains a comma.

3.10.8 String overflow from FUNCTION DISPLAY-OF mismanaged

The STRING statement could incorrectly manage overflow when input came from the FUNCTION DISPLAY-OF.

This has been corrected.

3.10.8 Runtime subscript checks compiling with -debug

Subscript runtime checking done when compiling with -debug is now done at every reference of the field.

3.10.8 D-ISAM internal sort error code management

When using D-ISAM, the internal sort managed error codes incorrectly in the case when the sort file did not exist.

This has been corrected.

3.10.8 VBISAM variable-size record calculations incorrect

When using VBISAM, READs and WRITEs of of variable-sized records could incorrectly calculate record size in some situations.

This has been corrected.

3.10.8 -O compiler flag fails to generate optimized code

The -O compiler flag could fail to generate code correctly in some complex sequences.

This has been corrected.

3.10.8 RECORD VARYING FROM X to Y anomaly

When using a RECORD VARYING FROM X to Y clause, and when the actual size of the level 01 record was greater that Y, the generated code could cause memory to be overwritten.

This has been corrected.

3.10.8 Breakpoint limitations on paragraph name

It was only possible to place one breakpoint on a paragraph name.

This has been corrected.

3.10.8 Using LINAGE with EXTFH

LINAGE was not supported when using EXTFH.

This has been corrected.

3.10.8 Using ROLLBACK with EXTFH

When used with EXTFH, files declared WITH ROLLBACK and not CLOSEd before the program ended did not have the ROLLBACK performed.

This has been corrected.

3.10.5 Parsing PIC 1 USAGE BIT variables in an IF statement incorrectly

The phrase IF FldA OR FldB, where FldA and FldB were described as PIC 1 USAGE BIT, was parsed incorrectly.

This has been corrected.

3.10.5 Optimizations not performed as expected inside nested IF statements

When optimizing using the -O compiler flag, it was possible for optimizations to not be performed as expected inside nested IF statements.

This has been corrected.

3.10.5 Rounding error on MOVE COMP-2 to USAGE DISPLAY

MOVEing data in a variable described as USAGE COMP-2 to a variable described as USAGE DISPLAY failed to round correctly when compiling with -O and -fround-fp.

This has been corrected.

3.10.5 Parsing single-character hexadecimal value representations

Single character hexadecimal value representations are now correctly right-parsed. As an example, the notation VALUE X'9' is right parsed as VALUE X'09'.

3.10.5 Long filename for target report file generates runtime error

In CITRW, when the file name of the target report file was too long, the runtime would generate an error.

This has been corrected.

3.10.5 -finitialize-opt failed when using nested programs

The -finitialize-opt failed to perform as expected when compiling nested programs.

This has been corrected.

3.10.5 WRITE AFTER/BEFORE ADVANCING ... on line sequential file anomaly

It was possible to get a <CR/LF> record delimiter when the line-seq-dos compiler configuration file flag was set to no.
This has been corrected.

Note-
When using WRITE AFTER/BEFORE ADVANCING, the record delimiter that is chosen depends on the setting of the `line-seq-dos` flag in your compiler configuration file.
When set to yes, a record delimiter of <CR><LF> is applied.
When set to no (the default), a record delimiter of <LF> is applied.

3.10.5 SUN Sparc platform availability is delayed

COBOL-IT is updating the Sun Sparc server. As a result, the availability of Version 3.10 on the Sun Sparc server will be delayed.

3.10.5 PROGRAM-ID IS INITIAL clause ignored

In version 3.9.37, the IS INITIAL clause of the PROGRAM-ID was ignored.

This has been corrected.

3.10.5 XML Generate output anomaly

The output generated by the XML Generate statement was different when the COB_CONSOLE_CP compiler flag was set.

This has been corrected.

3.10.5 Fix XML GENERATE

When a PIC X field was empty, XML Generate did not generate a single space, and this did not conform with IBM behaviors.

This has been corrected. A space is now generated in this case.

3.10.5 Debugging in Developer Studio on HP/UX

In some cases, when running on HP/UX, there could be problems communicating with the Developer Studio Debugger.

This has been corrected.

3.9

3.9.38

New

3.9.36 WRITE Line Sequential record to a named pipe

WRITEing a LINE SEQUENTIAL record to a named pipe is now supported.

3.9.34 Support for Visual Studio 2015 in Windows

Visual C 2015 is supported in Windows.

3.9.33 REPLACE statement may now be used inside a copy book

The REPLACE statement indicates text replacements to be made prior to compiling the source until another REPLACE statement is encountered, or until the REPLACE OFF statement is encountered. The REPLACE statement may now be used inside a copy book.

3.9.33 -sysout= now supports dot '.' as a file name

The -sysout compiler flag now supports a file name of dot '.' . When dot '.' is used as a file name, sysout is sent to stdout. As an example:

-sysout=.,SEQ,100 produces output on stdout as a sequential record of 100 bytes..

3.9.33 -fdisplay-ibm

The -fdisplay-ibm compiler flag affects the output of the DISPLAY Statement for numeric fields to be more compatible with IBM mainframe.

3.9.27 -funstring-use-move

The -funstring-use-move compiler flag affects the behavior of the UNSTRING verb. When using the -funstring-use-move compiler flag, if the target of an UNSTRING INTO operation is described as PIC 9, then the operation will be performed using a MOVE operation instead of raw copy operation. Then rules defined by the move-picx-to-pic9 compiler configuration flag are used for conversion.

3.9.26 In debugger , display <LOW-VALUE> or <HIGH-VALUE> for edited field

When displaying the value of a field in the debugger for editing, low-values is displayed as <LOW-VALUE> and high-values is displayed as <HIGH-VALUE>.

3.9.26 Performance improvements for SEQUENTIAL/RELATIVE files

Performance improvements have been implemented for I/O operations on LINE SEQUENTIAL, SEQUENTIAL, and RELATIVE files.

3.9.22 -fround-fp

The -fround-fp compiler flag controls the way COMP-1 or COMP-2 are "moved" into non-COMP-1 or COMP-2 target fields when the target field has fewer decimal places than the source field.

If the –fround-fp compiler flag is used, the value is rounded to the number of decimals of the target field. Otherwise, the value is truncated.

3.9.14 DISPLAY UPON PRINTER update

When used with a file defined with the environment variable COB_DISPLAY_PRINTER the DISPLAY statement performs OPEN and WRITE statements on the file, and the file is CLOSEd after each DISPLAY.

3.9.14 value-size-is-auto: [yes/no]

Default is `value-size-is-auto: no`

When set to yes, the CALL ..USING BY VALUE : default SIZE IS clause will be AUTO (current default is SIZE IS 4).

3.9.14 CITSORT –test command-line option

CITSORT now accepts the –test command line option. The -test option validates the command without actually executing the CITSORT command.

3.9.13 Report Writer supported on Windows

The COBOL-IT Report Writer is now supported in all supported Windows environments.

3.9.13 Optimization Enhancements

There have been a number of enhancements improving performance enhancements. These include:

- Performance enhancements when the –Os and –findex-optimize are used together.
- Index-Optimize and Decimal-Optimize are now set to YES when the -O or -O2 compiler flag is used. If you wish to use –O or -O2 compiler flags, and set either index-optimize and/or decimal-optimize to NO, then turn the index/decimal optimization off with the –fno-version of the compiler flag:
  As examples:
  >cobc –O –fno-index-optimize customer0.cbl
  >cobc –O –fno-decimal-optimize customer0.cbl
- The Decimal-Optimize: YES setting has been tuned, and now provides better performance improvement.

3.9.13 Evaluating expressions in Debugger supported in Windows.

The evaluation of expressions in the Developer Studio Debugger is now supported in all Windows environments.

3.9.7 -fcarealia-sign

Use CA Realia sign coding for Usage Display

Digit	Sign Digit Character for:					
	Positively-signed values			Negatively-signed values		
	-fsign-ascii	-fsign-ebcdic	-fcarealia-sign	-fsign-ascii	-fsign-ebcdic	-fcarealia-sign

0	0(30)	{(7B)	0(30)	p(70)	}(7D)	0(30)
1	1(31)	A(41)	1(31)	q(71)	J(4A)	!(21)
2	2(32)	B(42)	2(32)	r(72)	K(4B)	"(22)
3	3(33)	C(43)	3(33)	s(73)	L(4C)	#(23)
4	4(34)	D(44)	4(34)	t(74)	M(4D)	$(24)
5	5(35)	E(45)	5(35)	u(75)	N(4E)	%(25)
6	6(36)	F(46)	6(36)	v(76)	O(4F)	&(26)
7	7(37)	G(47)	7(37)	w(77)	P(50)	'(27)
8	8(38)	H(48)	8(38)	x(78)	Q(51)	((28)
9	9(39)	I(49)	9(39)	y(79)	R(52)	)(29)

3.9.7 carealia-sign: [yes/no]

Default is carealia-sign: no

When set to yes, the carealia-sign compiler configuration flag causes signed (PIC S9) variables to be stored according to CA-REALIA sign storage conventions. For more detail, see –fcarealia-sign compiler flag documentation.

3.9.7 Default settings change for index-optimize and decimal-optimize

Default settings for the index-optimize and decimal-optimize compiler flags have been changed to :

index-optimize: no

decimal-optimize: no

In order to use these optimizations, you must add these settings to your compiler configuration file:

index-optimize: yes

decimal-optimize: yes

3.9.5 -fdebugdb

The –fdebugdb compiler flag, when used with –g, store alls debugging information into a file name <modulename>.dbd. Copy this file to the same location as the the object file .so or .dll. This will permit the runtime debugger to load the debugging information dynamically when needed.

This is different from debugdb=<filename> where you have to specify a unique Debug db for the whole project.

Equivalent to: debugdb:yes in config file

3.9.4 -fexpand-exec-copy

The –fexpand-exec-copy compiler flag causes the compiler to expand COBOL COPY statements inside EXEC … END-EXEC blocks. This applies to both EXEC SQL and EXEC CICS blocks.

3.9.4 -fcall-comp5-as-comp

The –fcall-comp5-as-comp compiler flag affects the behavior of the CALL statement. On little-endian platform (intel Linux, Windows) when a call USING clause contains a literal , the –fcall-

comp5-as-comp compiler flag causes the literal to be copied as a COMPUTATIONAL value, rather than as a COMP-5 value.

Example: In the statement:

CALL “subprogram” USING 1234.
The literal 1234 is passed as a COMP value when using –fcall-comp5-as-comp.
Otherwise, the literal 1234 is passed as a COMP-5 value.

3.9.1 move-high-low-to-displaynumeric [error/zero/value]

Default is move-high-low-to-displaynumeric:value
ID: 1146980380

The move-high-low-to-displaynumeric compiler configuration flag affects the behavior of
MOVE HIGH-VALUES TO (usage display numeric data item) *and*
MOVE LOW-VALUES TO (usage display numeric data item).

When set to

error: trigger error at compilation "Invalid MOVE statement"
zero: move zeroes to display numeric item
value: move high- or low-values to display numeric item

3.9.0 cobcdb region interface available in IDE 1.7.19

Cobcdb enhancements that allow for the debugging of multiple regions in a single debugging session are now available in the Developer Studio. This provides enhanced ability in the COBOL-IT Debugger operating in the Developer Studio.

3.9.0 -findex-optimize

The –findex-optimize compiler flag optimizes indexing operations in statements with OCCURS clauses. As an example, consider the usage of the follow code:

```
…
01 IxdA  PIC 999  USAGE DISPLAY.
01 IxdB  PIC 999  USAGE DISPLAY.
 …
MOVE FLD-ARRAY(IxdA, ixdB) TO …
MOVE FLD-ARRAY(IxdA, ixdB) TO …
IF (FLD-ARRAY(IxdA, ixdB) …
```

Use of the –findex-optimize compiler flag will benefit the performance of these MOVE and IF statements by keeping the actual value of the index in a binary C cache, thus avoiding conversion from DISPLAY (or COMP-3) to a binary value each time the index is evaluated in a statement.

3.9.0 -fdecimal-optimize

The –fdecimal-optimize compiler flag optimizes the conversion from DISPLAY/COMP-3 to binary values in COMPUTE statements. When several COMPUTE statement are in the same paragraph, the compiler will minimize the conversions from DISPLAY/COMP-3 to binary values for fields that are used (and not modified) in different statements in the same paragraph.

3.9.0 COB_DISPLAY_PRINTER

The COB_DISPLAY_PRINTER runtime environment variable defines a file that is appended to for each “DISPLAY UPON PRINTER” statement. Each “DISPLAY UPON PRINTER” statement

OPENs this file, WRITEs to the file, and then CLOSEs the file.

Fixes

*3.9.38 " and " not accepted inside comment beginning with *>*

Including the word "and" inside a comment beginning with *> could cause an error to be generated.

This has been corrected.

3.9.38 XML Generate output was different when using COB_CONSOLE_CP

The output from XML Generate could be altered when the COB_CONSOLE_CP environment variable was set.

This has been corrected.

3.9.37 COMPUTE including integer constant and FLOAT (COMP-2)

When the -O compiler flag was used, a COMPUTE statement that incuded an integer constant and a FLOAT (COMP-2) could produce incorrect results.

This has been corrected.

3.9.37 Usage -extfh=... and -sysout= .. together in a program that include the INITIAL attribute in the PROGRAM-ID did not work correctly : Fixed

3.9.36 Program hangs in debugger in HP-UX

On HP-UX, in some conditions, which could include accessing an Oracle database, executing a module compiled with -g could cause the debugger to hang the program.

This has been corrected.

3.9.36 Anomaly Comparing USAGE FLOAT with other (non-FLOAT) field

When compiling with the -O compiler flag, comparisons in an IF statement between a variable declared as USAGE FLOAT (COMP-2) and another field were not correctly evaluated.

This has been corrected. When comparing data items declared as USAGE FLOAT (COMP-2) with other fields not declared as USAGE FLOAT (COMP-2), an internal conversion converts the other fields to USAGE FLOAT for the evaluation of the comparison (as IBM does). This internal conversion is made whether using the -O compiler flag or not.

3.9.36 MOVE PIC X to PIC 9 anomaly

When displaynumeric-mf50 is set to yes, the MOVE of a PIC X field to a PIC 9 field, where the PIC X field was smaller than the PIC 9 field, would incorrectly fill the target PIC 9 with SPACES instead of ZEROs.

This has been corrected.

3.9.36 Anomaly Comparing PIC 9 to literal with value greater than PIC 9 can store

When displaynumeric-mf50 is set to yes, when a PIC 9 field was compared to a literal whose value was greater than what the PIC 9 field could store, the image of the literal was incorrectly truncated to the size of the PIC 9 field and an incorrect comparison result was returned.

This has been corrected.

3.9.33 UNSTRING DELIMITED ALL correction

When using the UNSTRING DELIMITED ALL clause with a delimiter of more than 1 character, repeated delimiters could be handled incorrectly.

This has been corrected. The UNSTRING DELIMITED ALL clause now correctly skips repeated delimiters.

3.9.32 Memory leak corrected

A memory leak could occur under certain conditions when using LOCAL-STORAGE.

This has been corrected.

3.9.32 Corrections made to ibm.conf

The ibm.conf compiler configuration file has been improved to better reflect IBM COBOL behaviors.

3.9.30 Additional enhancements have been applied to -O code generation

In controlled tests, some cases were recorded where the -O compiler flag did not produce the expected optimizations.

3.9.30 Improved mainframe compatibility in compute using COMP-5

Mainframe compatibility for COMPUTE statements using COMP-5 data items has been improved.

3.9.27 Anomaly when writing LINE SEQUENTIAL file larger than 2GB

A regression error was found in 3.9.26 that could cause an error when writing a LINE SEQUENTIAL file larger than > 2GB.

This has been corrected.

3.9.27 Reduction of size of compiled object on AIX, HP/UX

When a large Working-Storage section is defined, the AIX and HP/UX compilers could produce very large object files.

This has been corrected.

3.9.26 IBM computation rule conformance improved when using -fcompute-ibm

In some situations, when using the -fcompute-ibm compiler flag, it was possible to still get computational behaviors that were non-conformant with IBM computation rules.

This has been corrected.

3.9.26 Memory Leak when using CBL_ERROR_PROC and CBL_EXIT_PROC

Usage of CBL_ERROR_PROC or CBL_EXIT_PROC could cause a memory leak.

This has been corrected.

3.9.26 Runtime crash starting module compiled with -g

A complex set of circumstances could cause the runtime to crash at startup when the main compiled object was compiled with -g.

This has been corrected.

3.9.21 Rounding error on COMP-2 data item

In controlled tests of fixes to rounding errors on COMP-2 data items made in 3.9.20, some cases were reported where rounding errors still occurred.

This has been corrected.

3.9.20 Unexpected behavior when using -fperform-osvs

A case was detected where the perform behavior when using -fperform-osvs was not compliant with the expected behavior.

This has been corrected.

3.9.20 -fls-expand-tab flag ignored

In some situations, the -fls-expand-tab compiler flag was ignored by the compiler.

This has been corrected.

3.9.17 RPATH stored by compiler

Version 3.9 of the compiler internally stored a variable called RPATH=/opt/cobol-it-64/lib in the binaries of cobc and other utilities. This was a problem if the directory /opt/cobol-it-64/lib existed and you tried to use another version of the compiler.

This has been corrected. RPATH is no longer stored in the binaries of cobc and other utilities.

3.9.17 ZD was incorrectly handled as non-signed numeric by CITSORT

The field definition was incorrectly handled as a non-signed numeric by CITSORT.

This has been corrected. ZD is now handled as a signed numeric as described in the DFSort documentation.

3.8

3.8.36

New

3.8.36 COB_DEBUG_MODULES=<program-id1>:<program-id2>….

COB_DEBUG_MODULES is a list of program-ids, in which the entries are separated by a colon character ":". Adding the program-id of a program in your application to the list of COB_DEBUG_MODULES causes the debugger to break at the entry of that program.

This provides an alternative way to attach the debugger to a running process in cases where programs do not contain calls to "C$DEBUG", or where you do not have access to the remote attach interface in the Developer Studio.

3.8.36 -fcopy-default-leading

The –fcopy-default-leading compiler flag affect the behavior of the COPY REPLACING statement.

When using the –fcopy-default-leading compiler flag, and when using the ==xxx== notation in a COPY REPLACING statement, the LEADING phrase is assumed by default. The LEADING phrase indicates that only the LEADING characters identified will be replaced if they match text in the copy file.

3.8.36 -fall-external-link

Causes the targets of the CALL statement to all be assumed to be external-links.
This can improve performance at runtime by optimizing the resolution of the CALL statement.

3.8.36 -finitialize-opt

Enables some optimization in the handling of the INITIALIZE statement.
This can improve performance at runtime by optimizing the execution of INITIALIZE statements.

3.8.36 -frelativefile-bigendian

Causes the record header of relative files to be stored in BigEndian format.

3.8.36 copy-default-leading:[yes/no]

Default is copy-default-leading:no
When set to yes,
affects the behavior of the COPY REPLACING statement.

When copy-default-leading is set to yes and when using the ==xxx== notation in a COPY REPLACING statement, the LEADING phrase is assumed by default. The LEADING phrase indicates that only the LEADING characters identified will be replaced if they match text in the copy file.

3.8.36 all-external-link:[yes/no]

Default is all-external-link:[no]
When set to yes,

Causes the targets of the CALL statement to all be assumed to be external-links.
This can improve performance by optimizing the resolution of the CALL statement.

3.8.36 initialize-opt:[yes/no]

Default is initialize-opt:no
When set to yes,
Enables some optimization in the handling of the INITIALIZE statement.
This can improve performance at runtime by optimizing the execution of INITIALIZE statements.

3.8.36 relativefile-bigendian:[yes/no]

Default is relativefile-bigendian:no
When set to yes,
Causes the record header of relative files to be stored in BigEndian format.

3.8.33 Performance improvements handling debugger expressions in IDE

Enhancements have been added to allow the Debugger in the COBOL-IT Developer Studio to handle debugger expressions with better performance.

3.8.32 mqseries.symb added to distribution

The mqseries.symb file has been added to the distribution of the COBOL-IT Compiler Suite Enterprise Edition. mqseries.symb is located in the config folder located under the $COBOLITDIR installation directory. This file includes all MQSeries symbol declarations for static link.

To include the mqseries.symb file with your default compiler configuration file, use the compiler flag: –conf=+mqseries.symb.

3.8.32 Debugger parsing of Field name updated for IDE 1.7.15

Compiler enhancements now allow the debugger to parse fully qualified data names. This provides enhanced ability in the COBOL-IT Debugger operating in the Developer Studio.

3.8.31 -frecord-depending-iso

The –frecord-depending-iso compiler flag causes a RECORD DEPENDING ON <FIELD> clause to be handled in an ISO-compatible manner. More specifically, the –frecord-depending-iso compiler flag causes files declared with RECORD DEPENDING ON <FIELD> clause, without any FROM or TO value, to assume a FROM and TO value of the maximum record size.

3.8.31 record-depending-iso:[yes/no]

Default is record-depending-iso:no

When set to yes,
The –record-depending-iso compiler configuration flag causes a RECORD DEPENDING ON <FIELD> clause to be handled in an ISO-compatible manner. More specifically, files declared with RECORD DEPENDING ON <FIELD> without any FROM or TO value to assume a FROM and TO value of the maximum record size.

3.8.30 CitSORT supports the FS data description in the FIELDS statement.

The FIELDS statement supports the FS data description. Requires CitSORT version 2.39.

FS	PIC S9.99 USAGE DISPLAY

Example:

```
> CITSORT USE PRES3.TXT ORG LS RECORD F 25 SORT FIELDS (1,2,FS,D) GIVE
PRES4.TXT ORG LS OUTREC FIELDS=(1,2,FS,' ',3,21)
```

3.8.30 CitSORT supports TO= syntax in OUTREC FIELDS.

CitSORT now supports the TO= syntax for OUTREC FIELDS.
The TO= syntax is used for converting data from an existing data format to a new format in the OUTREC FIELDS clause. Most commonly, it is used to re-format binary data into a printable format for purposes of reporting. Requires CitSORT version 2.39.

Example:
OUTREC Statement, Reformatting output record, converting data with TO=
This sample illustrates syntax for converting a numeric data item to FS data format using the TO= clause.

```
> CITSORT USE PRESIDENTS.DAT ORG SQ RECORD F 85 SORT FIELDS (1,2,NU,D) GIVE
PRES3.TXT ORG LS OUTREC FIELDS=(1,2,NU,TO=FS,' ',3,21)
```

3.8.30 Crash dump now stores the PID of the running process

The Memory crash dump now displays the PID of the runtime session. The PID is displayed at the beginning of the dump. See example below.

Cobol memory dump
++++++++++++++++++

PID: 6384
PROGRAM ID : crash_1 (C:\COBOL\CobolIT\Samples\crash\crash-1.cbl)
Current line : C:/COBOL/CobolIT/Samples/crash/crash-1.cbl:28
- - - - - - - - - - - - - - - - - -
WORKING-STORAGE
. . .

3.8.30 Debugging different regions independently is now possible

It is now posible to debug different regions independently in a runtime environment integrated with IBM TX Series.

3.8.28 Debugging a COBOL program run from a multi-threaded monitor

It is now possible to debug a COBOL program run from a multi-threaded transaction processing monitor.

3.8.28 line-seq-notrunc:[yes/no]

Default is line-seq-notrunc:no

The line-seq-notrunc compiler configuration flag affects the behavior of the runtime when a line sequential record is read that is longer than the declared record length.

When set to yes, the part of the record that exceeds the declared record length is returned as the next record.

When set to no, (the default), the record is truncated.

3.8.28 -ffast-op

Default is off

The –ffast-op compiler flag causes fast operations to be performed on numeric DISPLAY/COMP-3 data items. The –ffast-op compiler flag was introduced in version 3.8.7. It is no longer necessary to use the –fno-fast-op compiler flag to disable this functionality.

3.8.22 move-spaces-to-comp3:[error/space/zero]

Default is move-space-to-comp3: zero

The move-space-to-comp3 compiler configuration flag affects the behavior of the compiler and runtime when SPACES are moved to a COMP-3 data item.

The default behavior is for the compiler to issue an error: Error: Invalid MOVE statement

When set to space, the compiler does not issue an error. At runtime, SPACES are moved to the COMP-3 data item.

When set to zero, the compiler does not issue an error. At runtime ZEROES are moved to the COMP-3 data item.

3.8.22 COB_DEBUG_ALLUSER=1

The COB_DEBUG_ALLUSER environment variable, when set to 1, and when defined before running a COBOL program, causes the pipes that are created by the debugger to communicate with cobcdb to have Read/Write attributes for all users.

For the case where cob_init(. . .) has already been called, the same effect can be achieved by calling:

cob_debug_acl_alluser(rtd,1);

This will also ensure that the pipes that are created by the debugger to communicate with cobcdb have Read/Write attributes for all users.

3.8.22 Debugging code containing REPLACING clause

Debugging code where the REPLACING cause is active is now possible.

3.8.21 -fmf-file-optional

Affects the file-status codes returned on files declared as OPTIONAL and OPEN in EXTEND.

The –fmf-file-optional compiler flag causes files declared as OPTIONAL and OPEN in EXTEND to return file-status code “05” if the file is created and file-status code “00” if the file exists. The –fmf-file-optional compiler flag improves consistency with Micro Focus behaviors.

Alternatively, the runtime returns file-status code 00 in either case.

The –fmf-file-optional compiler flag corresponds to setting mf-file-optional:yes in the compiler configuration file.

3.8.21 mf-file-optional:[yes/no]

Default is mf-file-optional:no

When set to yes,
Affects the file-status codes returned on files declared as OPTIONAL and OPEN in EXTEND.

The mf-file-optional:yes compiler configuration flag causes files declared as OPTIONAL and OPEN EXTEND to return file-status code “05” if the file is created and file-status code “00” if the file exists. The mf-file-optional:yes compiler flag improves consistency with Micro Focus behaviors.

When set to no,
Files declared as OPTIONAL and OPEN EXTEND return file-status code “00” in both the case where the file did not exist, and was created, and the case where the file did exist.

The –mf-file-optional compiler configuration flag corresponds to setting -fmf-file-optional compiler flag.

3.8.20 -fcmp-opt

The –fcmp-opt compiler flag activates optimizations when comparing literals with variables. The –fcmp-opt compiler flag is set by default. Use –fno-cmp-opt to disable the functionality.

The –fcmp-opt compiler flag corresponds to the compiler configuration flag :
cmp-opt :yes

3.8.20 cmp-opt:[yes/no]

Default is cmp-opt:yes

The cmp-opt compiler flag activates optimizations when comparing literals with variables. When set to no, this functionality is disabled.

3.8.20 -fcopy-exec-replace

The –fcopy-exec-replace compiler flag affects the behavior of the COPY REPLACING statement. When using the –fcopy-exec-replace compiler flag, and when a COPY REPLACING == xxx == statement is performed, text inside EXEC / END-EXEC blocks are also replaced if applicable.

3.8.20 -fobj-cit

The –fobj-cit compiler flag causes compiled object to be generated with a cit extension instead of .dll (windows) or .so (unix/linux). The COBOL-IT runtime recognizes the .cit extension as an executable extension. The default behavior of the CALL statement has been changed, so that in a CALL “myprog” statement, the runtime will look first for a compiled object with a .cit extension, before searching for a .dll (Windows) or a .so (Linux/UNIX).

3.8.17 End of record marker of MF RELATIVE files

When using the –fmf-relativefile compiler flag, the end-of-record marker for relative files is consistent with the setting of the compiler configuration flag line-seq-dos.

When line-seq-dos:yes, the end of record setting is CR/LF

When line-seq-dos:no, the end of record setting is LF

3.8.17 -fcopy-partial-replace

The –fcopy-partial-replace compiler flag increases the compatibility of the COPY REPLACING behavior with Micro Focus compiler behaviors.

When a pattern like COPY FIC1 REPLACING == WJXX- == BY == WJ03- == is processed :
If this flag is on, the preprocessor uses a partial replacement as defined by MF and ANSI2002 standard.
If it is off (the default) the IBM mainframe and ANSI85 standard is used.

The –fcopy-partial-replace compiler flag corresponds to the compiler configuration flag : copy-partial-replace :yes

3.8.17 initcall:<program-name>

The initcall compiler configuration flag names modules to be called immediately before the first statement of a program is executed. The initcall compiler configuration flag corresponds to the initcall compiler flag.

3.8.17 CBL_DEBUGBREAK

CBL_DEBUGBREAK is a synonym for C$DEBUG. CBL_DEBUGBREAK is a library routine which can be called using either the PID of the runtime session, or the value of the environment variable COB_DEBUG_ID. For more details, see the documentation of the C$DEBUG library routine.

3.8.17 EC-SIZE-ZERO-DIVIDE:[yes/no]

Default is EC-SIZE-ZERO-DIVIDE:no

When set to yes,
Then, in combination with trap-unhandled-exception:yes will capture all division by zero error events if no ON SIZE ERROR clause is present.

3.8.12 Support for Visual C 2013

On Windows platforms, the Visual C 2013 Compiler is now supported.

3.8.12 No more support for Visual C 2005

On Windows platforms, Visual C 2005 is no longer supported.

3.8.12 Handling cases where "." is missing at the end of a field declaration

The COBOL-IT parser now accepts some cases where a "." is missing at the end of a field declaration. This enhancement improves compatibility with the Micro Focus parser.

3.8.12 More information now stored in DebugDB

More information is now stored in DebugDB. As a result, the object size of a debug version is smaller.

3.8.12 Field-A.Field-B.Field-C alias of Field-C IN Field-B in Field-A.

The COBOL-IT parser accepts ***FIELD-A.FIELD-B.FIELD-C*** as an alias of ***Field-C IN Field-B In Field-A*** . This enhancement improves compatibility with the Micro Focus parser.

3.8.12 Handling of SYNC statement in a table element below an OCCURS clause

When a SYNC statement was used in an element defined below an OCCURS statement, the size of the OCCURS group is now adjusted to ensure that all elements respect the SYNC.
As an example :

```
01 GRP.
 03 A.
   05 A10 occurs 10.
      10 A11 PIC X.
      10 A12 PIC S9(08) COMP SYNC.
      10 A13 PIC X(05).
```

Previously the memory map would be

```
*******    0 :   130 :      -      :  A
*******    0 :    13 :    1 -    10 :    A10
*******    0 :     1 :      -      :     A11
*******    4 :     4 :      -      :     A12
*******    8 :     5 :      -      :     A13
```

With the problem that A12(2) would not be aligned unless the SYNC statement.

Now the map size of A10 is adjusted to ensure SYNC of all element A12 and the map is :

```
*******    0 :   160 :      -      :  A
*******    0 :    16 :    1 -    10 :    A10
*******    0 :     1 :      -      :     A11
*******    4 :     4 :      -      :     A12
*******    8 :     5 :      -      :     A13
```

3.8.12 move-picx-to-pic9: mf40

Default is move-picx-to-pic9: cit
Defines the runtime behaviour when moving alphanumeric (PIC X(n) USAGE DISPLAY), to non-signed display numeric data items (PIC 9(n) USAGE DISPLAY).

When set to “mf40’’
The source field is copied, right-justified. Upper half-bytes are replaced with 0x30. As a result, a MOVE PIC X(3) VALUE “ABC” to a PIC 9(3) field stores “123” in the PIC 9(3) field. This is the Micro Focus 4.0 default behavior.

3.8.12 constant: "key=value"

Provides a way to define constants that can be tested for purposes of conditional compilation. When using the constant “key=value” compiler flag, the conditional compilation below will test true.

```
$if key=value
$else
$end
```

3.8.12 makesyn: oldvalue=newvalue

Provides a way to make a reserved word a synonym for another reserved word. The first word, represented by “oldvalue” becomes a synonym of the second word, represented by “newvalue”. A common usage is to make COMP a synonym of COMP-5. To do this, set:

```
makesyn: comp=comp-5
```

The makesyn "oldvalue=newvalue" compiler flag provides compatibility with the MAKESYN directive. A COBOL verb or field-name may be used as “old-value”. The COBOL-IT compiler will replace all instances of this “old-value” with the “new-value” when compiling.

CAUTION- While this provides an equivalent capability to the implementation of the MAKESYN directive in other COBOLs, the order of the parameters is reversed. COBOL-IT requires that the “old-value” be listed first, and followed by the “new-value”.

3.8.12 -fmf-relativefile

The –fmf-relativefile compiler flag causes the runtime to assume the Micro Focus format for relative files for both READ and WRITE operations.

3.8.12 mf-relativefile :[yes/no]

Default is mf-relativefile :no
Allows for compatibility with Micro Focus relative files.

When set to yes, the COBOL-IT runtime assumes the Micro Focus format for relative files for both READ and WRITE operations.

3.8.12 COB_FILE_RELATIVE_MF=Y

The COB_FILE_RELATIVE_MF environment variable, when set to Y, causes the COBOL-IT runtime to assume the Micro Focus format for relative files for both READ and WRITE operations.

3.8.9 -debugdb=<filename>

When compiling with debug, the cit compiler use to store all the Debugging meta information into the program binaries. This could make debugging program very huge. And in some situation event prevent the program to load in memory.

Now with compilation option –debugdb=<filename> the compiler store the debugging meta data into a SQLite3 database .

The same data base should be used for the complete project.
Then when debugging the debugger runtime will use the database with the path given at compilation time or if defined use the COB_DEBUGDB environment variable contents.

3.8.8 –debugdb=<DebugDB-name>

Causes debugging meta data to be stored into the <DebugDB-name> file.
At runtime you must set the runtime COB_DEBUGDB to allow the runtime to read the debugger metadata.:

>export COB_DEBUGDB=<DebugDB-name>
Currently only 1 database may be used at a time. As a consequence, the customer must use the same database for all of the programs in a run unit. Metadata from multiple programs may be stored in the single database.

Data is stored in an SQLITE3 database. You may wish to experiment with tools available online for manipulating data in SQLITE3 databases.

3.8.7 -ffast-op, -fno-fast-op compiler flags

Default is On.

The –ffast-op compiler flag causes fast operations to be performed on numeric DISPLAY/COMP-3 data items.

The –fno-fast-op compiler flag causes these optimizations to be disabled.

Fixes

3.8.36 null-param default is listed as yes

When source files were compiled with –debug, list files incorrectly reported the default of the null-param compiler configuration file setting as yes.

This has been corrected. The null-param default is now correctly listed as no.

3.8.33 Anomaly on SYNC of an array that requires fewer than 8 bytes

It was possible for the SYNC operation to behave incorrectly when used on an array that required fewer than 8 bytes.

This has been corrected.

3.8.32 Package build fixed for HPUX

An anomaly in the HPUX package build was detected.

This has been corrected.

3.8.31 READ NEXT lock management failure

In Version 3.7, VB/ISAM files could fail to properly manage locks on READ NEXT operations in some situations.

This has been corrected.

3.8.30 full-cancel:yes updates cob_enterprise_cancel_region

When used with full_cancel:yes, cob_enterprise_cancel_region is now updated to ensure a module is not used in another region before unloading it.

3.8.30 Loop calling cob_tidy could crash runtime

Programs calling cob_tidy in a loop could crash the runtime after about 40 loops.

This has been corrected.

3.8.30 relax-bounds-check:yes did not allow invalid offset to be specified

When relax-bounds-check was set to yes, reference-modified variables were not evaluated.
In the case below, as an example, a variable is declared as PIC X(10), and <variable>(11:1) is referenced. With relaxed-bounds-check:yes, this was considered to be an error.

This has been corrected. The invalid offset is not accepted.

3.8.28 Error when using FCD in 64 bits

The File Control Description (FCD) is the structure through which information passes to and from EXTFH-compliant data sources. An FCD is created for each file that is mapped to an EXTFH-compliant data source.

In some cases, the 64-bit runtime did not update the fcd_line_count field in the FCD.

This has been corrected.

3.8.28 EVALUATE ...WHEN [condition...] NOT parsing error

In some cases, a WHEN NOT condition could be incorrectly parsed.

This has been corrected.

3.8.28 DELETE error on RELATIVE FILE

It was possible for a DELETE statement to incorrectly delete the next record instead of the last record read when moving sequentially through a RELATIVE FILE.

This has been corrected.

3.8.28 Comment marker not recognized when mf_comment:no

When the compiler configuration file variable mf-comment:no was set, a comment marker in column 7 could be ignored by the compiler in some cases.

This has been corrected.

3.8.28 Error on ADD/SUBTRACT with COMP-3 fields

In some cases, ADD and SUBTRACT statement containin data items declared with USAGE COMP-3 could produce incorrect results by returning the wrong sign.

This has been corrected.

3.7

3.7.63

New

3.7.47 Debugging a COBOL program run from a multi-threaded monitor

It is now possible to debug a COBOL program run from a multi-threaded transaction processing monitor.

3.7.41 COBOL Memory Dump output includes memory address for each field

The output of the COBOL Memory Dump has been enhanced by adding the memory address of each field.

3.7.41 CIT EXTFH resolves ASSIGN target names using environment variables

When called to open a file, the CIT EXTFH handler assumes the COBOL Configuration File setting of Filename-mapping:yes, which enables the full set of file aliasing for resolving data file names. File aliasing is handled through the use of environment variables, which include:

Set the environment variable DD_DATAFILE to [filename]
Set the environment variable dd_DATAFILE to [filename]
Set the environment variable DATAFILE to [filename]
Locate [filename] in a directory named by the COB_FILE_PATH environment variable

3.7.38 COBCTMP=<path>

The COBCTMP=<path> compiler environment variable, when defined, changes the directory in which temporary files are stored.

3.7.33 fcompute-ibm/fno-compute-ibm

Default is –fno-compute-ibm.

Important note: The fcompute-ibm compiler flag was used by default between versions 3.7.20 and 3.7.32. If your application relies on this behavior, you must re-compile using the –fcompute-ibm compiler flag, or set compute-ibm:yes in your compiler configuration file.

The fcompute-ibm compiler flag causes arithmetic expressions (like a+B*c) in COMPUTE statements, and comparisons to use IBM COBOL defined rules for determining the number of decimals used in intermediate results. For details, please consult IBM documentation.

The fno-compute-ibm compiler flag causes the maximum number of decimals (machine-dependent, up to 37) to be used in results of arithmetic expressions in COMPUTE statements and comparisons.

3.7.33 compute-ibm:[yes/no]

Default is compute-ibm:no

Important note: Between versions 3.7.20 and 3.7.32, the default was compute-ibm:yes

If your application relies on this behavior, you must use the –fcompute-ibm compiler flag, or set compute-ibm:yes in your compiler configuration file.

When set to yes, causes arithmetic expressions (like a+B*c) in COMPUTE statements, and comparisons to use IBM COBOL defined rules for determining the number of decimals used in intermediate results. For details, please consult IBM documentation.

When set to no, causes the maximum number of decimals (machine-dependent, up to 37) to be used in intermediate results of arithmetic expressions in COMPUTE statements and comparisons.

3.7.26 cobc –V (in Windows) reports Microsoft version used in build of cobc

The command "cobc –V" now reports the Microsoft C Compiler version is reported as version XXYY where XX is the major version of the Microsoft C Compiler, and YY is the first minor version of the Microsoft C Compiler. As an example, see the output of cobc –V below :

```
C:\COBOL\CobolIT>cobc -V
cobc (COBOL-IT) Enterprise
Version 3.7.26 (32 bits)
Build date Aug 19 2014 09:08:39
Build with Microsoft(c) Compiler version 1600
```

Note that Microsoft C Compiler Versions do not match Visual Studio versions.
To cross reference this information with Visual Studio version information, see the table below.
To check the version of the Microsoft C Compiler being used by COBOL-IT, enter the command:

Visual Studio	Cl version	Visual Studio	Cl version
8.0 (2005)	1400	10.0 (2010) SP1	1600
9.0 (2008)	1500	11.0 (2012)	1700
9.0 SP1	1500	12.0 (2013)	1800
10.0 (2010)	1600		

You can check the version of the Microsoft C Compiler being used by COBOL-IT in your COBOL-IT command shell. As an example, see the output of the cl command below:

```
C:\COBOL\CobolIT>cl
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 16.00.40219.01 for 80x86
Copyright (C) Microsoft Corporation.  All rights reserved.
```

3.7.20 Performance improved on Arithmetic Operations using integer literals

The performance of the ADD, SUBTRACT MULTIPLY and DIVIDE verbs has been improved in certain cases. For the case where a numeric field is described with a decimal notation, and contains an integer value, with no decimals, (for example, PIC 99V99 VALUE 10.), the performance of arithmetic operations using that numeric field have been improved.

To activate this optimization you must use the -O compiler flag, and set the compiler configuration file flag :
binary-truncate: no
or
notrunc:yes

*3.7.20 Preprocessing of lines with leading **

It was possible, when compiling source in terminal (-free) format, for the COBOL-IT preprocessor to treat the leading asterisk of an exponent (**) operator as a comment indicator when the exponent operator was the first token on the line.

This has been corrected.

Note- When encountering a comment line containing a leading * the COBOL-IT preprocessor now renders the leading comment * as *> when compiling source written in ansi (-fixed) source format, and renders the leading comment * as **> when compiling source written in terminal (-free) source format.

3.7.10 INSPECT verb performance improved on AIX, Linux platforms

The performance of the INSPECT verb has been improved on AIX, and Linux platforms.

3.7.10 synchronized-double-word-bound:[yes/no]

Default is synchronized-double-word-bound:no

The synchronized-double-word-bound compiler configuration flag affects the alignment of data items declared as USAGE BINARY, which also contain the SYNC clause.

The default behavior is consistent with the behaviors of the IBM and Micro Focus compilers. Specifically, when the SYNC clause is used, binary fields are aligned either on 2- or 4-byte boundaries, depending on the size of the field.

Fields of 1-2 bytes in size are aligned on 2-byte boundaries.
Fields greater than 2-bytes in size are aligned on 4-byte boundaries.

When set to yes, binary fields are aligned on 2- or 4- or 8-byte boundaries, depending on the size of the field. This was the default COBOL-IT behavior prior to version 3.7.9.

Fields of 1-2 bytes in size are aligned on 2-byte boundaries.
Fields of 3-4 bytes in size are aligned on 4-byte boundaries.
Fields greater than 4 bytes in size are aligned on 8-byte boundaries.

3.7.8 mf-hostnumcompare:[yes/no]

Default is mf-hostnumcompare:no

The mf-hostnumcompare:yes compiler configuration entry provides compatibility with Micro Focus in the case where USAGE DISPLAY numeric data items are initialized, and the HOSTNUMCOMPARE directive is used.

When set to no (the default), a numeric USAGE display data item (for example, a PIC 99 data item) that is initialized with defaultbyte=0, or LOWVALUE, and compared to any numeric value would be considered smaller, since LOWVALUE is considered smaller than any numeric value.

When set to yes, a numeric USAGE display data item (for example, a PIC 99 data item) that is initialized with defaultbyte=0 or LOWVALUE is considered to be initialized to ZERO.

3.7.8 fmf-hostnumcompare

The -fmf-hostnumcompare compiler flag causes a numeric USAGE DISPAY data item that is initialized with defaultbyte=0 or LOWVALUE to be considered to be initialized to ZERO.

3.7.8 SYNC clause on group item propagates to all elementary items

The SYNC clause, when set on a group item now causes the SYNC clause to be set for all sub-fields of the group. This enhancement provides greater compatibility with Micro Focus.

3.7.0 Linux Debugger does not use signals

On Linux platforms, the debugger no longer uses signals. With this enhancement, the debugger is able to avoid IO interruptions.

3.7.0 User running cobcdb not same as user running the runtime process

With this enhancement, the debugging pipes are created using the rules of the unix command umask. If before running the cobol program, you set a umask to 0000, any user will be able to

debug the process.

3.7.0 Runtime part of the debugger now run in a separate thread

With this enhancement, the Debugger GUI (Deet or Eclipse) continues to be available even when the process is active. While the process is running, the Debugger can read a field value, for example. Breakpoints can be set when the runtime process is paused on an ACCEPT statement.

3.7.0 Cobsetjmp more accurately emulates Micro Focus behaviors

The cobsetjmp function has been enhanced to better emulate Micro Focus behaviors.

3.7.0 The GMP library is statically built in the libcob library

With this enhancement, conflicts are avoided when using a newer version of gcc (newer than 4.6), which also embeds a copy of libgmp.

3.7.0 ICU is bult with cit prefix

With this enhancement, conflicts are avoided with the platform-provided version of the ICU library.

Fixes

3.7.63 Correction to displaynumeric-mf50 behaviors

When displaynumeric-mf50 is set to yes:

- The MOVE of a PIC X field to a PIC 9 field, where the PIC X field was smaller than the PIC 9 field, would incorrectly fill the target PIC 9 with SPACES instead of ZEROs.

- When a PIC 9 field was compared to a literal whose value was greater than what the PIC 9 field could store, the image of the literal was incorrectly truncated to the size of the PIC 9 field and an incorrect comparison result was returned.

These have been corrected.

3.7.61 Rounding problem expressing COMP-2 with exponent (**) operator

It was possible to get a rounding error when using a COMP-2 variable with an exponent operator.

This has been corrected.

3.7.61 Space causes unexpected results expressing negative numeric-edited picture

It was possible to get unexpected results when inserting a space between a "-" and a numeric picture digit in a numeric edited picture string.

This has been corrected.

3.7.61 Unexpected behavior using -fperform-osvs

In some complex cases, it was possible that -fperform-osvs could return an unexpected behavior.

This has been corrected.

3.7.43 PROGRAM-ID had to match case with program in COB_FULL_CANCEL

When the COB_FULL_CANCEL environment variable was used, the program name in the program-id had to be identical, and have the same case, as the program name used in the CANCEL statement. If these program names were not in the same case, the runtime did not properly release

the binaries.

This has been corrected.

3.7.43 MOVE PIC X with LOW-VALUE to PIC ZZ9 not interpreted as MOVE 0

An alphanumeric field (PIC X) with LOW-VALUE, when MOVEd to a numeric edited (PIC ZZ9) field, would not interpret the binary zeros as ZERO. A subsequent DISPLAY of the numeric edited field would not display the 0.

As further clarification, moving a PIC X data item containing LOW-VALUE to a field described as USAGE COMP-3 or USAGE COMPUTATIONAL or to a numeric edited field, is like moving ZERO.

This has been corrected.

3.7.43 'FILLER$1' not level 01 or 77 compilation error

In rare cases, a compilation error containing a token, such as 'FILLER$1' not level 01 or 77 could be returned.

This has been corrected.

3.7.42 –ffcdreg compiler flag requires –use-extfh compiler flag

Using –ffcdreg requires that –use-extfh also be used, but including the –ffcdreg compiler flag without the –use-extfh compiler flag would not generate an error.

This has been corrected.

3.7.42 Windows Only : Memory Leak using Microsoft LoadLibrary/Free Library

When a program loaded COBOL module (.dll) using the Microsoft LoadLIbrary API, and then unloaded the module using the Microsoft FreeLibrary API, the FreeLibrary API did not free all of the allocated memory.

This has been corrected.

3.7.42 Line Sequential file with multiple record definitions truncates data

When a LINE SEQUENTIAL file had several record definition of different size, data could be truncated incorrectly on a READ statement. More specifically, in the case where the first record written to the file uses the record definition with the shorter size, and this is followed by a READ of the file using the record definition of the longer size, the data returned by the READ statement would be truncated to to the length of the record with the shorter size.

As an example:

```
SELECT FILEA LINE SEQUENTIAL…
FD FILE
01 RECA .
  03 ID      PIC 99.
  03 Data PIC X (100).
01 RECB  .
  03 ID      PIC 99.
  03 Data PIC X (10).
```

..

```
OPEN OUTPUT FILEA
WRITE RECB
CLOSE FILEA
…
OPEN INPUT FILEA
READ FILEA
…
```

The data returned by the READ of FILEA was truncated to the size of RECB.

This has been corrected.

3.7.41 C++ comments could generate compilation errors

C++ comments (// instead of /* */) in distributed .h files could generate compilation errors on AIX in some situations.

These comments have been removed.

3.7.41 COBOL Memory dump (function cob_dump_debug_info) more robust

The Cobol Memory dump (function cob_dump_debug_info) could generate a SIGSEGV error if a linkage pointer were invalid.

This has been corrected.

3.7.40 Debugger does not stop on breakpoint

Due to a regression introduced in the debugger in version 3.7.39, in certain cases, the debugger did not stop correctly on breakpoint.

This has been corrected.

3.7.40 Problem accessing memory dump from "C"

When the runtime did a memory dump (post mortem dump) or cob_dump_debug_info was called from C, invalid LINKAGE fields could be appended that could cause the COBOL-IT runtime to crash.

This has been corrected.

3.7.40 OCCURS DEPENDING can cause C compilation error

Locating the DEPENDING field of an OCCURS DEPENDING clause in LOCAL STORAGE, and compiling with the –g (debugging) compiler flag could produce a C compiler error.

This has been corrected.

3.7.39 Incorrect debugger line position

The debugger line could be positioned incoreectly when parsing an EXEC SQL INCLUDE xxxx END-EXEC statement.

This has been corrected.

3.7.39 WRITE AFTER 0 anomaly

In some situations, the WRITE AFTER 0 statement did not insert a CR, when this was the expected behavior.

This has been corrected.

3.7.39 Temporary files in /tmp not erased after compile

In some cases, temporary files created in the /tmp directory were not erased after the compile operation was completed.

This has been corrected.

3.7.39 Accept <Screen_name> auto behavior corrected

The AUTO attribute on ACCEPT <Screen-name> did not create the expected behavior.

This has been corrected.

3.7.39 Screen Section: Justified attribute is now correctly handled

The JUSTIFIED attribute, when used in the Scren Section, did not create the expected behavior.

This has been corrected.

3.7.38 Reduced performance for FILE I/O operatons on small records

A regression error introduced in version 3.6 could cause reduced performance for FILE/IO operations on small records in Sequential and Line Sequential Files.

This has been corrected.

3.7.38 Windows Only : Memory Leak using Microsoft LoadLibrary/Free Library

When a program loaded COBOL module (.dll) using the Microsoft LoadLIbrary API, and then unloaded the module using the Microsoft FreeLibrary API, the FreeLibrary API did not free all of the allocated memory.

This has been corrected.

3.7.38 Compilation error on CALL Statement

Compiling with the –Os compiler flag, a CALL <module-name> statement, where <module-name> was also the name of a COBOL variable could result in a C compilation error.

This has been corrected.

3.7.38 Debugger error after compiling with -preprocess

When –preprocess was used with some CICS preprocessor, the debugger did not always show the correct file.

This has been corrected.

3.7.35 Upper Case & Lower case working with NATIONAL fields

The LOWER-CASE and UPPER-CASE intrinsic functions, when used with fields described as USAGE NATIONAL, could produce incorrect results.

This has been corrected

3.7.35 Intrinsic Function “integer-part” in COMPUTE statement

ID: 1146617371

In some situations, the usage of the intrinsic function “integer-part” in a COMPUTE statement could produce incorrect results.

This has been corrected.

3.7.35 COMPUTE statement with edited field results

ID: 1146618582

When the target of a COMPUTE statement was a NUMERIC EDITED data item, incorrect results could be produced in some situations.

This has been corrected.

3.7.33 Problems using Deet fixed on Windows and AIX

Some anomalies were reported using Deet in the Windows and AIX operating environments.

These have been corrected.

3.7.32 INITIALIZE verb anomalies when compiling for optimization

When the –O or –O2 compiler flags were used, the INITIALIZE verb would incorrectly data items described as FILLER and USAGE POINTER to LOW-VALUE.

This has been corrected.

3.7.32 Internal Sort Statement fix using very large files

When very large files were sorted using the internal Sort statement, unexpected results could occur.

This has been corrected.

3.7.32 EC-DATA-INCOMPATIBLE:yes setting causes runtime error

ID: 1146590630

When the compiler configuration flag EC-DATA-INCOMPATIBLE:yes was used with the -foptimze-move compiler flag, a runtime error could be generated in some situations, when the source field was declared in a REDEFINES clause.

This has been corrected

3.7.32 INITIALIZE statement anomaly

When compiling with the –O2 compiler flag, the INITIALIZE statement could cause a data item to be INITIALIZED incorrectly. The case for PIC 1 USAGE BIT fields was corrected in version 3.7.31. This correction fixes other cases. To integrate this fix, programs must be recompiled.

This has been corrected.

3.7.31 FILLER described as PIC 1 BIT initialialized that should not be initialized

When compiling with the –O2 compiler flag, the INITIALIZE statement could cause a data item described as FILLER PIC 1 BIT to be initialized, when it should not be initialized.

This has been corrected.

3.7.31 Precision used for COMPUTE statement incorrect

The implementation of the –fcompute-ibm compiler flag in version 3.7 behaved incorrectly in some cases, when the definition of the target field of a COMPUTE statement was less precise than the definition of one or more numeric literals used in the COMPUTE statement.

This has been corrected.

3.7.26 VBISAM incorrect file error "10" on READ NEXT

ID: 1146487436

When multiple users were accessing the same file, VBISAM could incorrectly return an error "10" in some situations.

This has been corrected.

3.7.26 Memory management crash calling cob_enterprise_cancel_region

Calling cob_enterprise_cancel_region many times could cause a memory access violation.

This has been corrected.

3.7.26 Output of internal (COBOL) SORT is empty file

In some conditions, the internal COBOL SORT statement could produce and empty file as output.

This has been corrected.

3.7.26 Comments passed through –preprocess optimized

When using –preprocess, COBOL-IT's intermediate file handling can cause additional comments to be added to the file passed to the preprocessor, for purposes of debugging. In some situations, this could cause problems with procob.

This has been corrected.

3.7.20 COPY REPLACING Statement produces incorrect results

ID: 1145754652

The COPY *copylib* REPLACING *old-text* BY *new-text* Statement could produce incorrect results in some cases when *old-text* was a string that contained SPACES.

This has been corrected.

3.7.20 COMPUTE ROUNDED error

ID: 1145830624

When the target variable of a COMPUTE statement was described as a numeric edited field, with PIC --,---,---. , the COMPUTE ROUNDED statement could produce incorrect results.

This has been corrected.

3.7.20 COB_SCREEN_INPUT_REVERSED only works on monochrome terminals

ID: 1146310142

On color terminals, COB_SCREEN_INPUT_REVERSED does not create the REVERSED effect.

The COB_SCREEN_INPUT_REVERSED environment variable only works on monochrome terminals. This has been clarified in the documentation.

3.7.20 COB_SCREEN_INPUT_UNDERLINED does not work

ID: 1146310168

Setting COB_SCREEN_INPUT_UNDERLINED=Y did not change the default behavior.

This has been corrected.

3.7.20 COB_SCREEN_INPUT_INSERT_TOGGLE cannot be reset

ID: 1146310190

After setting COB_SCREEN_INSERT_TOGGLE to Y, re-setting the environment variable to its default setting of N had no effect.

This has been corrected.

3.7.20 COB_SCREEN_INPUT_BOLDED=Y does not work

ID: 1146310194

Setting COB_SCREEN_INPUT_BOLDED =Y did not change the default behavior.

This has been corrected.

3.7.20 COB_SCREEN_RAW_KEYS does not work

ID: 1146311214

Setting COB_SCREEN_RAW_KEYS=Y did not change the default behavior, and did not allow crt-status values to be returned when the documented "raw keys" were pressed.

This has been corrected.

3.7.20 COB_SCREEN_ESC requires COB_SCREEN_EXCEPTIONS

ID: 1146311262

Setting COB_SCREEN_ESC=Y did not change the default behavior, and did not allow crt-status values to be returned when the escape key was pressed.

You are required to set COB_SCREEN_EXCEPTIONS=Y, in order to enable the COB_SCREEN_ESC=Y environment variable setting. This has been clarified in the documentation.

3.7.20 CALL "SYSTEM" USING [string containing X"00"] aborts compiler

ID: 1146467300

On the zLinux platform, concatenating a low-value byte (X"00") to the end of a file name, and using the file name in a CALL "SYSTEM" could cause the compiler to abort.

This has been corrected.

3.7.20 Memory Leak loading a COBOL-IT object file

ID: 1146471026

In certain situations, a memory leak could occur when using COBOL-IT API routines to CALL COBOL-IT from "C".

This has been corrected.

3.7.20 SIGN SEPARATE LEADING returning negative value when no sign entered

ID: 1146473038

A data element described with SIGN SEPARATE LEADING could return a negative value when no sign was entered.

This has been corrected.

3.7.20 deet interface does not display

ID: 1146492490

The deet interface did not display, after compiling a source file –g, and running the command: >deet [filename]

This has been corrected.

3.7.20 Unable to detect UP & DOWN arrow keys on ACCEPT [Screen]

ID: 1146495276

When operating inside a Screen Section, it was not possible to return the CRT-STATUS code of the UP and DOWN arrow keys on an ACCEPT [Screen] statement.

This has been corrected.

3.7.20 Moving literals to data items with USAGE COMP / COMP-4 / COMP-5.

ID: 1146495698

In some cases, the optimize-move:yes setting in the compiler configuration file, when used with the binary-truncate:yes setting in the compiler configuration file, could cause unexpected results when moving literals to data items described as USAGE COMP, USAGE COMP-4, and USAGE COMP-5. In these cases, it was necessary to reset to optimize-move:no, and sacrifice some performance.

This has been corrected.

3.7.20 Compiler does not allow REDEFINES of variable-length table

ID: 1146496090

In some cases, the odo-slide:no setting did not cause the compiler to allow a REDEFINES of a variable-length table. In these cases, the compiler would return an error:
Error: The original definition '[variable-name]' cannot be variable length

This has been corrected.

3.7.20 Exponentiation sign not reproduced in COPY/REPLACING

ID: 1146500234

When an exponentiation sign (**) was included in the [old text] of a COPY/REPLACING phrase, it was not reproduced in the [new text], producing an incorrect result.

This has been corrected.

3.7.14 Memory Leak on CALL

When a large number of CALL operations were performed, a memory leak could occur.

This has been corrected.

3.7.13 Bus Error (core dumped) on Solaris

ID: 1146347476

Under certain conditions, the COMPUTE statement could generate a Bus Error (core dump) on the Solaris platform.

This has been corrected.

3.7.13 Using exponential notation "**" on a new line causes compiler error

ID: 1146442074
ID: 1146481428

In some situations, a COMPUTE statement with an exponential notation ("**") described on multiple lines, in which the exponential notation was placed at the beginning of a line, would cause the compiler to abort.

Example:
COMPUTE end-result = 2
**3.

This has been corrected.

3.7.12 Problem using + without space in LINE/COL clauses in the SCREEN Section

ID : 1146309324

In the SCREEN SECTION, incrementing LINE/COLUMN numbers using the + (integer increment) notation, and not placing a space before (integer increment) would cause the (integer increment) to be interpreted as the LINE or COLUMN number. Other COBOLs treat this condition as a syntax error.

This has been corrected.

3.7.12 Problem with DISPLAY following DISPLAY/ACCEPT

ID : 1146309456

After the DISPLAY/ACCEPT of a SCREEN described in the SCREEN SECTION, a field-level DISPLAY could produce unexpected results.

This has been corrected.

3.7.12 Compiler crashes when using LIKE Clause and –fas400-like

ID : 1146350810

The –fas400-like compiler flag could cause the compiler to abort in certain cases, in which the program used the LIKE clause.

This has been corrected.

3.7.12 COMP-3 containing non-numeric data aborts runtime when not referenced

ID : 1146435700

A data item described as USAGE COMP-3 containing SPACES could cause a program to abort when it was not explicitly referenced.

When using DEFAULT-BYTE, it was possible for the runtime to perform IS NUMERIC tests at the end of a paragraph on fields that were not explicitly referenced.

This has been corrected.

3.7.12 Sequential, Line Sequential and Relative file corruption (Windows Only)

In some circumstances, the occurrence of the character sequence 0x0A in a data string could be handled incorrectly when being written to a Sequential, Line Sequential, or Relative file, causing corruption of the data.

This has been corrected.

3.7.12 CITSORT of Sequential / Line Sequential Files

In some circumstances, CITSORT could produce unexpected results when sorting Sequental or Line Sequential files.

This has been corrected.

3.7.11 Cob_enterprise_cancel_region anomaly with reload of new version of module

When using the cob_enterprise_cancel_region function-

After a full cancel, the application could load a new version of a module (.so / .dll) once, but after that, a subsequent full cancel did not unload the module in memory, so it was not possible to load a new module again.

This has been corrected in version 3.7.11. The full cancel always unloads the module in memory, and new versions of a module may be loaded subsequently. Version 3.7.11 is fully compatible with 3.7.10 and no recompilation is required.

3.7.10 INSPECT verb causes crash on SUN SPARC platform

On the SUN SPARC platform, in certain situations, the INSPECT statement could causes a program to abort.

This has been corrected.

3.7.10 cob_enterprise_cancel_region function does not free memory

The cob_enterprise_cancel_region function did not free the memory allocated by CBL_ALLOC_MEM.

This has been corrected.

3.7.8 COMPUTE Statement containing comp-2 field produces incorrect result

ID: 1146390836

In certain situations, storing the results of a COMPUTE statement in a data element described as USAGE COMP-2 could produce incorrect results.

This has been corrected.

3.7.8 Infinite loop condition on creation of Core Dump

Under certain conditions, it was possible to create an infinite loop in the creation of the Core Dump.

This has been corrected.

3.7.8 Copy "XFHFCD.CPY" did not work when source in free format

The COPY "XFHFCD.CPY" clause, used with the EXTFH interface, did not work when source files were stored in free format.

This has been corrected.

3.7.4 Handling of relative files in AIX

In some situations, relative file handling on the AIX platform could produce unexpected results.

This has been corrected.

3.7.4 Unresolved CALL statements in AIX

In some situations, a CALL statement could be reported as “not resolved” incorrectly.

This has been corrected.

3.7.2 Debugger aborts when using a table of PIC 1 USAGE BIT data items

ID: 1146336600

The usage of a table of PIC 1 USAGE BIT data items could cause the debugger to crash in some situations.

This has been corrected.

3.7.2 Larger-refefines-ok:no could not be applied to an 01-level data item

ID: 1146339538

The compiler configuration flag setting larger-redefines-ok:no could produce unexpected results when a redefines was made on an 01-level data item.

This has been corrected.

3.7.2 Cancel produces unexpected results when more than 1 region is active

It was possible to get unexpected results using CANCEL and CANCEL_REGION when more than one region was active.

This has been corrected.

3.6

3.6

New

3.6 COBOL-IT Compiler Suite Enterprise Edition Dual Mode (ASCII/EBCDIC):

COBOL-IT now provides support for data encoded in EBCDIC format on all UNIX/LINUX platforms with the COBOL-IT Compiler Suite Enterprise Edition Dual Mode.

The COBOL-IT Compiler Suite Enterprise Edition Dual Mode provides users with the ability to manage all internal data storage in EBCDIC format, and process data files encoded in EBCDIC format on open UNIX/LINUX operating systems with the addition of the **–febcdic-charset** compiler flag.

The COBOL-IT Compiler Suite Enterprise Edition Dual Mode extends COBOL-IT's value by providing the ability to offload batch programming portions of an Enterprise Application hosted on a Mainframe. The COBOL-IT Compiler Suite Enterprise Edition Dual Mode can be used to perform the batch FILE I/O operations on data files encoded in EBCDIC format.

The following scenario would be one example of how to use the COBOL-IT Compiler Suite Enterprise Edition Dual Mode, and reduce valuable CPU cycles:

- In a Mainframe batch processing cycle with EBCDIC-encoded data files, processing can now be transferred to a Mainframe/Unix share folder accessible by the COBOL-IT Compiler Suite Enterprise Edition Dual Mode. As an example:
    - A Mainframe Batch stream COPYs unconverted EBCDIC-encoded input files to a Mainframe/Unix share folder.
    - The Mainframe Batch programs that process the EBCDIC-encoded input files are compiled and run using the COBOL-IT Compiler Suite Enterprise Edition Dual Mode. Data is transformed, per the instructions in the Batch programs, and results written to EBCDIC-encoded output files in the Mainframe/Unix share folder.
    - The EBCDIC-encoded output files are copied from the Mainframe/Unix share folder to their original location on the Mainframe.
    - Mainframe processing continues.

The COBOL-IT Compiler Suite Enterprise Edition Dual Mode is available on all Open Systems UNIX or LINUX (ASCII) platforms and has the following features:

When using the **–febcdic-charset** compiler flag:

- The COBOL-IT Compiler Suite Enterprise Edition Dual Mode compiles source files encoded in ASCII format.
- The COBOL-IT Compiler Suite Enterprise Edition Dual Mode makes the following conversions automatically:
    - All string literal values found in the source code are converted by the compiler to EBCDIC format.
    - All string literals expressed in Hexadecimal format (using the notation VALUE X "00" for example) are presumed to be hexadecimal notations expressing literals in EBCDIC format.
        - The compiler flag -fsign-ebcdic is assigned as a default. As a result, all USAGE DISPLAY numeric literals are stored in using EBCDIC sign formats

- All figurative constants, (for example SPACE, ZERO, LOW-VALUES, etc), are converted by the compiler to EBCDIC format.

Runtime Features:

- **Data Storage**
  - The internal storage of the values of all data items is EBCDIC-encoded.
  - The target of the VALUE Clause of all USAGE DISPLAY variables is always presumed to be in EBCDIC format
- **Data Handling**
  - All string operations are performed on EBCDIC values
- **File Handling**
  - In file I/O operations that are designed to READ, and WRITE/REWRITE data, no conversions are performed on the data. Input files are presumed to contain data that is in EBCDIC format. Since no conversions are performed, the data in the resulting output file will also be in EBCDIC format.
  - In file I/O operations that are used to locate a data file, as in an OPEN, CLOSE, DELETE, for example, the file name is converted to ASCII before checking for the named file in the host file system.
- **Calling Subprograms**
  - In CALL statements, the subprogram name that is the target of the CALL statement is converted to ASCII format before checking for the named file, or symbol, in the host file system.
  - In a CALL "SYSTEM" statement, the program name that is the target of the CALL "SYSTEM" statement is converted to ASCII format before checking for the named executable in the host file system.
- **Console Handling**
  - In a DISPLAY [target] UPON CONSOLE statement, [target] is converted to ASCII format before the DISPLAY upon the console.
  - Runtime Error messages that are displayed on the console are converted to ASCII format before the DISPLAY upon the console.
  - In an ACCEPT [target] FROM CONSOLE statement, [target] is converted from ASCIII to EBCDIC before being stored as a data value.

Limitations and know issues

At run time, File I/O operations do not make any conversions. As a consequence, no mixture of ASCII and EBCDIC data is permissible in the input file. Input files must contain data that is entirely stored in EBCDIC format.
At run time, the parameters in a CALL … USING [param1, param2…] statement are not converted. If an external "C", or system routine is called with parameters, then a "C" wrapper needs to be written to convert the parameters from EBCDIC format to ASCII format before calling the external "C" or system routine.
The Screen Section is not managed.
EXEC … END-EXEC statements are not managed and would probably not work correctly if the underlying library expects data to be stored in ASCII format.

3.6 fsources-codepage <codepage-id>

The –fsources-codepage <codepage-id> defines the code page to be used when editing the source and the code page used for string literals in the COBOL source code.

3.6 fcodepage <codepage-id>

The –fcodepage <codepage-id> compiler flag defines the encoding of PIC X in memory.
If –fsources-codepage is specified, the compiler converts from the codepage-id used in the –fsources-codepage compiler flag to the codepage-id used in the –fcodepage compiler flag.

The debugger makes use of the –fcodepage <codepage-id> compiler flag setting, and converts alphanumeric (PIC X) data to/from UTF-8 when sending data to/from GUI interface.

The conversion of data described with USAGE NATIONAL to PIC X also uses the –fcodepage <codepage-id> compiler flag setting. If –fcodepage <codepage-id> is not specified then if –fsources-codepage <codepage-id> is specified, the setting of –fsources-codepage is used for the conversion of data described with USAGE NATIONAL to PIC X.

3.6 febcdic-charset

The –febcdic-charset compiler flag requires a dedicated license.
When using the –febcdic-charset compiler flag, the COBOL-IT Compiler and Runtime store and manage data in the EBCDIC encoding format. Source code is stored in ASCII format.

The –febcdic-charset compiler flag causes modules to be created that apply the EBCDIC character set to file operations, and internal data storage. When using the –febcdic-charset compiler flag, all of the programs in a runtime unit must be compiled with the –febcdic-charset compiler flag. If a main entry program that is not compiled with the –febcdic-charset compiler flag CALLs a program that is compiled with the –febcdic-charset compiler flag, or conversely, if a main entry program compiled with the –febcdic-charset compiler flag CALLs a program that is not compiled with the –febcdic-charset compiler flag, the CALL will fail, and the COBOL-IT runtime will abort.

3.6 fmainframe-vb

The –fmainframe-vb compiler flag causes WRITEs and READs of Variable Blocked files to assume formats compatible with the Mainframe Z/OS Cobol Format.

3.6 COB_CONSOLE_CP=<codepage-id>

The COB_CONSOLE_CP=<codepage-id> runtime environment variable, when defined, causes the DISPLAY UPON CONSOLE/DISPLAY UPON SYSOUT phrases to convert the codepage defined using the –fcodepage compiler flag to <codepage-id>.
As an example (on a Linux X Console):

```
>cobc –x myprog.cob –fcodepage latin1
export COB_CONSOLE_CP=UTF-8
./myprog
```

Will correctly display literals encoded in latin1 on the console.

3.6 File-oriented Environment Variables evaluated on OPEN statement

The following runtime environment variables are now evaluated when the OPEN statement is executed by the runtime. Changes can be made during the runtime session, allowing users to apply different settings to the same file, or different settings to different files : For more informationon the usage of these environment variables, see the COBOL-IT Reference Manual.

COB_SYNC
COB_LS_NULLS
COB_LS_DOS
COB_LS_FIXED
COB_FILE_TRACE
COB_PAD_BUG
COB_VAR_REC_PAD

Fixes

3.6 Developer Studio debugger display anomaly

In some situations, the Developer Studio debugger could incorrectly display a literal.

This has been corrected. The UTF-8 protocol now used by the Developer Studio debugger allows for the correct display of PIC X (non-national) fields.

3.6 Incorrect DISPLAY of PIC 99V99 showing 5 characters instead of 4

ID: 1146092920

In some cases, the BLANK WHEN ZERO attribute on a NUMERIC EDITED field (as an example, PIC 99V99), could create a DISPLAY anomaly which would produce an incoreect DISPLAY of the value of the variable.

This has been corrected.

3.5

3.5.8

New

3.5 COB_PAD_BUG=1

The runtime environment variable COB_PAD_BUG=1 must be used in version 3.5.8, when:

- Using a variable length RECORD SEQUENTIAL file (REC MODE "V") created in a COBOL-IT version prior to 3.5.8 that is greater than 2GB in size AND
- Using the compiler configuration flag variable-rec-pad-mf: yes .

When the user had set variable-rec-pad-mf: yes, then when a record sequential file exceeded 2GB in size, all records whose last character address was larger that 2GB(0x80000000) were written with an additional 4-byte low-value filler. This anomaly did not occur when record addresses were smaller than 2GB (0x80000000).

Prior to version 3.5.8, this extra padding was not detected by the COBOL-IT runtime, and did not affect the handling of the data in the file. However, the extra 4-byte low-value filler did cause errors when using external tools, such as Syncsort.

With the correction of this problem, the COBOL-IT runtime will detect the extra padding condition, and these files will not be READable if the environment variable COB_PAD_BUG=1 is not set.

Note that while setting the environment variable COB_PAD_BUG=1, the old behavior of adding extra padding when writing records whose last character address is larger that 2GB is preserved, and the ability to read these records is preserved.

To rebuild a file that has been corrupted in this manner, set the environment variable COB_PAD_BUG=1, use the COBOL configuration file setting variable-rec-pad-mf:yes, and run a program which READs through the file with records containing extra padding, and WRITEs the records back out to a flat file with non-variable length (REC MODE "F"). Then, set the environment variable COB_PAD_BUG=0, use the configuration file setting variable-rec-pad-mf:yes, and run a second program which READs the flat file, computes the length of the variable-length record, and and WRITEs the records back out to a flat file with variable length (REC MODE "V"). Note that two separate programs are required, as COBOL-IT reads the COB_PAD_BUG environment variable just once when the runtime starts.

3.5 Compatibility of NATIONAL-OF and DISPLAY-OF intrinsic functions enhanced

ID: 1146106244

The compatibility of the NATIONAL-OF and DISPLAY-OF intrinsic functions with other COBOL implementations of these intrinsic functions has been enhanced.

Substrings notations are now supported. For example,
MOVE FUNCTION DISPLAY-OF(NATIONAL-VAR(1:4) 13488) TO STRING-VAR.

NATIONAL-OF for codepages 13488 and 1200 is always Big-Endian.

NATIONAL-OF ("X") can be used in the target clause of an INSPECT… CONVERTING TO statement. For example :
INSPECT NATIONAL-VAR CONVERTING SPACE TO FUNCTION NATIONAL-OF("X")

3.5 RECORD VARYING FROM 1 supported in FD

ID: 1145927210

In the FD, the syntax

RECORD MODE IS VARIABLE
RECORD VARYING FROM 1

is now supported by the compiler.

3.5 ICU library used to manage code pages in NATIONAL functions

COBOL-IT has replaced the library used to manage code-pages in NATIONAL functions. Prior to version 3.5, the Iconv library of functions was used. Beginning with version 3.5, the ICU library (http://site.icu-project.org/home) is used. As a consequence:
- Code pages now have the same names on Unix, Linux, and Windows platforms.
- Many more code pages are supported
- EBCDIC encoding is now supported

3.5 –list-codepage

The –list-codepage compiler flag is an informational compiler flag that lists all supported codepages in the following format:
Codepage: [list of synonyms for codepage]

As an example:

UTF-8 : UTF-8 ibm-1208 ibm-1209 ibm-5304 ibm-5305 ibm-13496 ibm-13497 ibm-17592 ibm-17593 windows-65001 cp1208 x-UTF_8J unicode-1-1-utf-8 unicode-2-0-utf-8

3.5 –check-codepage <name>

The –check-codepage <name> compiler flag is an informational compiler flag that you can use to check if a given codepage is recognized by the ICU library.

As examples:

To check for support of codepage 500, (which is supported):

C:\COBOL\CobolIT>cobc -check-codepage 500

500 is found as ibm-500_P100-1995

To check for support of codepage 13488 (which is not supported):

C:\COBOL\CobolIT>cobc -check-codepage 13488

Internal Error: can't open converteur 13488 : U_FILE_ACCESS_ERROR

Fixes

3.5 Correction on 32-bit installation for Windows for VC 2005 & 2008

In some cases, the installation script for the 32-bit Windows compiler could produce incorrect results, when the Visual C 2005 or Visual C 2008 selection was made.

This has been corrected.

3.5 Odo-slide default set to No in various compiler configuration files

The default setting of odo-slide in the compiler configuration files ibm.conf, mvs.conf, cobol85.conf and cobol2002.conf files was incorrectly set to Yes.

This has been corrected.

3.5 FUNCTION LENGTH and TRIMR return 1 instead of 0

ID: 1146126298

The FUNCTION LENGTH and TRIMR functions could incorrectly return a 1 instead of a 0 if the field being evaluated had a length of 0 bytes.

This has been corrected.

3.5 Warning/syntax error lines incorrect and debugger prompt off as well

ID: 1146130586

When using the Developer Studio Debugger, the debugger in the IDE could identify warning and syntax error lines incorrectly, such that the line numbers given did not match the source. In these situations, the debugger prompt could be misrepresented as well.

This has been corrected.

3.5 Null pointer in Linkage Section could cause Eclipse Debugger to crash

In some conditions Eclipse Debugger could crash when a field in the Linkage section used was described as USAGE POINTER, and was not assigned a value. In this case, the POINTER would have a NULL value, and this could cause problems in some cases.

This has been corrected.

3.5 The command "DISPLAY ERASE EOS" aborts the compiler

ID: 1146110382

In some situations, the statement “DISPLAY ERASE EOS” could cause the compiler to abort.

This has been corrected.

3.5 GO-TO Tags are not recognized by the Compiler

ID: 1146108480

When the target of a GO TO statement was a tag in the for [99]E[99] where [99] represents a numeric string, the Compiler would mistake the tag for a real number (COMP-2), and return an error.

As an example-

```
PROCEDURE DIVISION.
…..
     IF END-OF-FILE GO TO 20E1.
…..
20E1.  EXIT.
```

This has been corrected.

3.5 INSPECT TALLYING on National fields produce wrong results

ID: 1146039650

When a PIC N field was the target of a TALLYING clause in an INSPECT statement, it was possible for incorrect results to be returned.

This has been corrected.

3.5 Incorrect REWRITE on variable-length C-TREE record

ID: 1146056106

In some situations, the C-Tree EXTFH interface could REWRITE a variable length record incorrectly.

This has been corrected.

3.5 COBMF anomaly in Windows

ID: 1146061266

When using cobmf in Windows, the command line provided a reference to the citextfh_dll.lib, but did not also provide a means for the compiler to locate that library.

This has been corrected.

3.5 Debug comment too long

ID: 1146106302

When using the Developer Studio Debugger, and when compiling with the –preprocess, and –g compiler flags, the COBOL-IT compiler generates comments in the source, in the form :

DEBUGxxx GUBED"file name ....."

These compiler-generated comments would be truncated if they extended past column 72.

This has been corrected. Compiler-generated comments are split over several lines if needed, with the split occurring at column 72.

3.5 Call to x"AF" to set TAB as a function key not working as expected

ID: 1146103622

The CALL x”AF” implementation of COBOL-IT has been enhanced to be more compatible with the implementation of Micro Focus. Behaviors associated with setting a TAB key as a function key if AUTO is used on a field control have been refined for purposes of compatibility.

3.4

3.4.19

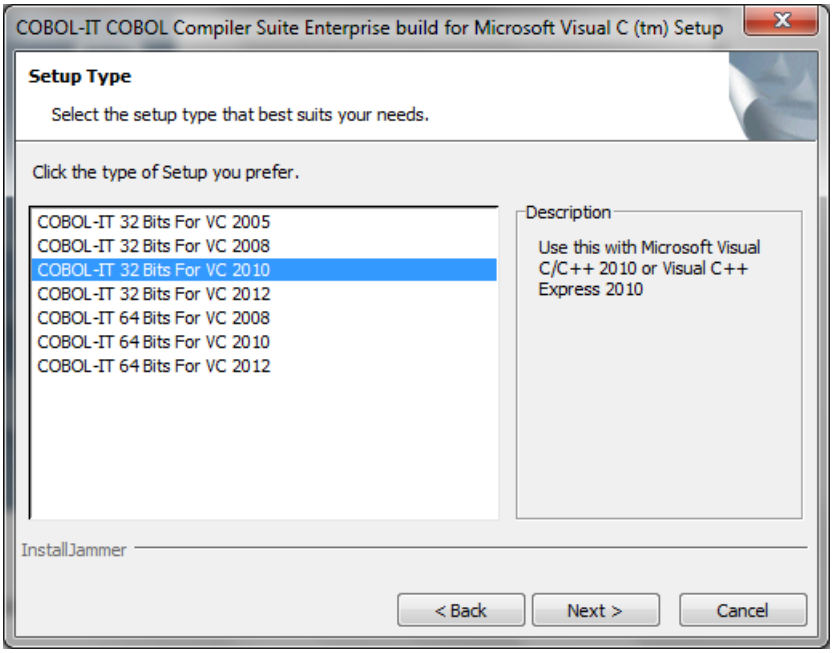
New

3.4 Windows Visual C 2012 is now supported

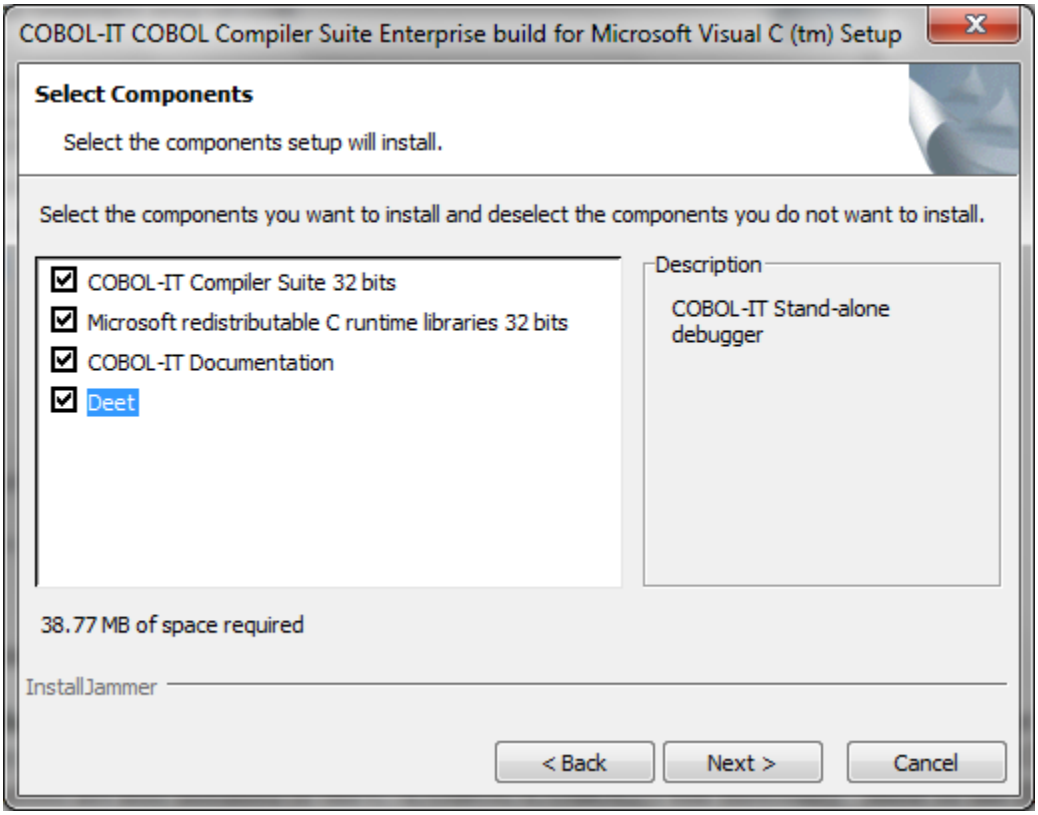
The COBOL-IT Compiler Suite Enterprise Edition is now available for the 64-bit Microsoft Visual C/C++ 2012 or Visual C++ Express 2012 compiler.

3.4 Windows distribution allows pairing of VC Compiler at setup time

The Windows Installer now provides the ability to indicate, at setup time, whether you wish to install a 32-bit or 64-bit Windows, compiler, and which VC compiler you wish to pair it with.



3.4 Windows distribution allows for selection of components at setup time



Selecting the Deet component causes the stand-alone debugger "deet" to be installed. Deet is executed by running "deet.bat", which is installed in C:\COBOL\COBOL-IT\bin. Required DLLs and executables are also installed in the C:\COBOL\COBOLIT\bin directoriy. Most of the files required for the operation of "deet" are installed in the C:\COBOL\COBOLIT\lib directory, or in subdirectories of that library.

Selecting the Documentation component causes C:\COBOL\ COBOLIT \doc folder to be installed. Note that the CitSORT and COBOL-IT Compiler Suite Reference Manuals are now included in this directory at time of installation.

Selecting the Microsoft redistributable files causes the Microsoft Visual C/C++ runtime library to be installed, which is necessary if Visual C is not installed. The Visual C/.C++ runtime library files are installed in the C:\COBOL\COBOLIT\bin directory, and include:

Microsoft.VC80.CRT.manifest
Microsoft.VC90.CRT.manifest
msvcm80.dll
msvcm90.dll
msvcp100.dll
msvcp110.dll
msvcp80.dll
msvcp90.dll
msvcr100.dll
msvcr110.dll
msvcr80.dll
msvcr90.dll
vccorlib110.dll

Selecting the COBOL-IT Compiler Component causes the remaining files to be installed. In the installation directory, these include setenv_cobolit.bat, and uninstall.exe.

The core files in the C:\COBOL\COBOLIT\bin directory are:

bdbextfh_dll.dll
cat.exe
citextfh_dll.dll
CITMAKE.EXE
citsort.exe
cobc.exe
cobcdb.exe
cobcrun.exe
cobmf.exe
libcobit_dll.dll
libdb47.dll
libdb53.dll
libiconv2.dll
libintl3.dll
md5deep.exe
uname.exe
vbcheck.exe
vbrecover.exe

The core installation also includes:

C:\COBOL\ COBOLIT \config-	Compiler configuration files used by COBOL-IT
C:\COBOL\ COBOLIT \copy-	Copy files used by COBOL-IT
C:\COBOL\ COBOLIT \include-	Include files used by COBOL-IT
C:\COBOL\ COBOLIT \include\libcob-	Include files used by COBOL-IT
C:\COBOL\ COBOLIT \lib-	Core COBOL-IT libraries

3.4 CitSORT distinguishes between DOS/Unix in Line Sequential Organization

CitSORT has added a variation of the org statement, for Line Sequential files in DOS environments:

org lsd

The **org lsd** notation causes lined sequential files to be created with record terminators of CR/LF.

Note that line sequential files in UNIX/Linux environments should continue to use the existing notation of :

org ls

The **org ls** notation causes line sequential files to be created with record terminators of LF.

3.4 CITSORT_LS_DOS

The environment variable CITSORT_LS_DOS forces record terminators for all line sequential files to be CR/LF. For existing installations, where CitSORT is being used in a DOS environment, and where scripts use an **org ls** notation, this would be required when updating to version 3.4.18, as the behavior of CITSORT has changed, and by default, the **org ls** notation causes record terminators of LF to be created.

3.4 CitSORT supports multi-processing

Citsort now has the ability to split the sort process into several tasks to take advantage of multicore systems.

The parameter:

-processmax =	<integer>	defines the maximum number of processes to use. This must be defined to enable multi-processing
-processrec =	<integer>	defines the minimum number of records written before creating a new process

3.4 Support-debugging-line:[ok/error]

Default is support-debugging-line:ok

The support-debugging-line compiler configuration entry provides a way to override the compiler's support for usage of the character "D" in column 7 to mark a debugging line.

When set to ok (the default) , source lines that contain a "D" character in column 7 are ignored, unless the compiler configuration flag debugging-line:yes is set, in which case the line is compiled.
.
When set to error, the compiler generates an error when it encounters a "D" character in column 7.

3.4 Improved compatibility of Extfh handling of RECORD CONTAINS clause

When an FD contains a RECORD CONTAINS xx CHARACTERS clause, such as:
RECORD CONTAINS 70 CHARACTERS
but the actual record size described has fewer characters than are named in the clause, COBOL-IT now sends the stated number of characters in the RECORD CONTAINS clause for both MIN-RECORD-SIZE and MAX-RECORD-SIZE.

Note that in the case above, when using VBISAM, which does not rely on EXTFH handling, COBOL-IT would detect the smaller actual record size, and pass it through as the MIN-RECORD-SIZE.

This behavior increases compatibility with the EXTFH implementation of the c-Tree ISAM file system. If you encounter an EXTFH-compliant file system that requires that the non-EXTFH default behavior described above, then you may compile with the

-fno-strict-record-contains compiler flag, or set :

strict-record-contains:no in the compiler configuration file .

3.4 strict-record-contains:[yes/no]

Default is strict-record-contains:yes

The strict-record-contains compiler configuration entry affects the handling of the RECORD CONTAINS xx CHARACTERS clause in file systems using then EXTFH interface, when the actual record size has fewer characters than are named in the clause.

When set to yes (the default):
COBOL-IT sends the number of characters stated in the clause as both MIN-RECORD-SIZE and MAX-RECORD-SIZE.

When set to no:
COBOL-IT detects the smaller actual record size, and passes it through as the MIN-RECORD-SIZE, whie passing the number of characters stated in the clause as the MAX-RECORD-SIZE.

Note that the strict-record-contains compiler configuration flag has no effect on VBISAM files, as they do not use the EXTFH interface.

3.4 -fshare-all-default

The -fshare-all-default compiler flag causes all files to be declared implicitly as SHARE WITH ALL.

-fshare-all-default ***may be*** used in conjunction with :

-fshare-all-autolock Causes all files with a SHARE WITH ALL clause in their SELECT statement to be declared implicitly as LOCK MODE IS AUTOMATIC.

-fshare-all-manulock Causes all files with a SHARE WITH ALL clause in their SELECT statement to be declared implicitly as LOCK MODE IS MANUAL.

-fshare-all-default ***should not be*** used in conjunction with:

-fexclusivelock Causes all files with no LOCK MODE clause in their SELECT statement to be declared implicitly as LOCK MODE is EXCLUSIVE.

-fmanuallock Causes all files with no LOCK MODE clause in their SELECT statement to be declared implicitly as LOCK MODE is MANUAL unless a SHARING clause in the SELECT statement or in the OPEN statement indicates otherwise.

3.4 -fctree-no-full-qualification

ID: 1145808238

The –fctree-no-full-qualification compiler flag affects the behavior of the –fgen-xdd compiler flag, causing it to not generate the fully qualified data names in the XDD description of the file. When using the –fctree-no-full-qualification compiler flag with the –fgen-xdd compiler flag,

the field name generated is :
xxx_<Field_Name>

Where:
xxx is a unique number (position in the structure),
<Field_Name> is the name of the data field, without any other prefix or suffix

3.4 CitSORT SORT-EBCDIC clause

ID: 1145784300

CitSORT supports the SORT-EBCDIC clause.

SORT-EBCDIC causes record-sequential ASCII files to be sorted in EBCDIC order.

As an example:

```
citsort fields (4,26,CH,A)
SORT-EBCDIC
use FILE1
        org ls record (v 1 32000)
give FILE2
```

3.4 CitSORT supports SORT of compressed indexed files

ID: 1145797134

CitSORT has added a variation of the org statement:

```
org ix1
```

to represent the case where indexed files that are compressed are being sorted.
When data is compressed in an indexed file, you must describe the organization as ix1, and you must declare all indexes.

As an example:

```
citsort copy
        use DATAFIL
            org ix1 key=(1,18, P, 291,10, AD) record f 1100
        give DATAFILsort
        org sq record f 1100
```

3.4 Identifier-length: <max-length>

ID: 1145795806

Default is identifier-length: 0

The identifier-length compiler configuration flag allows compatibility to be achieved with other compilers, as regards the maximum length of a variable name.

When set to 0
COBOL-IT has no limit on the length of a variable name, and will never test for this, or generate a warning.

When set to a positive, non-zero value
COBOL-IT will generate a warning if the length of a variable name exceeds the the max-length named by identifier-length.

To achieve compatibility with the IBM COBOL compiler, for example, which generates a warning when a variable name exceeds 30 characters, set `identifier-length: 30.`

3.4 Redefine-identifier: [error / warning / ok]

ID: 1145722930

Default is redefine-identifier: ok

The redefine-identifier compiler configuration entry has been added to the "Dialect Features" component of the compiler configuration file. The redefine-identifier compiler configuration flag affects compiler behavior when ambiguous identifiers exist in the source code.

As an example, if your Working-Storage Section contained two identical declarations:

```
77 field-1  pic x(10).
77 field-1   pic x(10).
```

When set to ok
  The compiler does not generate an error.
When set to warning
   The compiler generates a warning
When set to error
   The compiler generates an error

3.4 fibm-sync (default is on), -fno-ibm-sync

The –fibm-sync compiler flag causes the SYNC attribute to be applied to a group item if the first elementary field in the group item is described with the SYNC attribute.
This is a default behavior.

To turn off this (default) behavior:
The –fno-ibm-sync compiler flag does not cause the SYNC attribute to be applied to a group item in the first elementary field in the group item is described with the SYNC attribute.

Fixes

3.4 ACCEPT FROM COMMAND-LINE running in Debug mode

ID: 1145983566
ACCEPT from COMMAND-LINE did not work, when running in the debugger.

This has been corrected.

3.4 COB_LS_NULLS not working correctly

ID: 1145989906
When the compiler configuration flag **line-seq-mf:no** was set, the environment variable COB_LS_NULLS had no effect.

This has been corrected. Setting the environment variable COB_LS_NULLS now overrides the setting of the compiler configuration file flag line-seq-mf, and line-seq-dos.

3.4 Errors reported compiling main "C" program containing CIT include files

ID: 1145931466
Syntax errors were reported when compiling a main "C" program containing COBOL-IT include files in both 32- and 64-bit Linux platforms, using COBOL-IT Compiler Suite Enterprise Edition version 3.4.14. The problem did not occur on other Linux platforms, and did not occur when using COBOL-IT Compiler Suite Enterprise Edition version 3.3.

The problem was attributed to a non ANSI-standard comment in cit_rtinterface.h.

The problem has been corrected.

3.4 END PROGRAM in source causes IDE/Deet debugger to fail

ID: 1145963064

A source file ending with the line END PROGRAM. Caused both Deet, and the Developer Studio Debugger operate as though the program were not compiled for debugging.

This has been corrected.

3.4 INSPECT REPLACING statement aborts compiler when keyword ALL omitted

ID: 1145897770

The INSPECT REPLACING statement requires that the keyword ALL be used. When the keyword ALL was omitted, the compiler would abort.

This has been corrected.

3.4 Error on CitSORT copy of empty indexed file to line sequential file

ID: 1145890256

The execution of a CitSORT COPY command from an empty indexed file to a line sequential file incorrectly generated an output file with one record initialized to low-values.

For reference, an example of the citsort copy command would be:
citsort copy use CUSTIDX record F 264 org IX key '(1,101,P)' give CUSTSEQ org LS

This has been corrected. The command now correctly generates an empty ouput file.

3.4 ADD Statement truncation anomaly when no ON SIZE ERROR present

ID: 1145855878

When the receiving field of an ADD statement did not contain enough digits to hold the resulting SUM of the ADD statement, the ADD statement could produce unexpected results.

As an example:

```
77  var-1  PIC 99  VALUE 99.
…
   ADD 1 TO var-1.
…
```

In this case, the value of var-1 was incorrectly being returned unchanged. In the example above, the correct behavior is to truncate the results of the ADD statement.

This has been corrected.

3.4 Support-Debugging-line not included in List File

ID: 1145880252

The support-debugging-line compiler configuration flag was not reported in the listing file.

This has been corrected.

3.4 Numeric comparisons on non-numeric data

ID: 1145858978

Documentation Clarification.

When numeric fields contain non-numeric data, numeric comparisons may not have the same behavior with COBOL-IT as with another COBOL compiler.

As an example, in the code below, COBOL-IT tests TRUE on IF VAR-1 = 0.

```
77  VAR-1    PIC 9(3) USAGE DISPLAY.
…
MOVE “@@@” to VAR-1.
…
IF VAR-1 = 0
    DISPLAY “VAR-1 = 0”
ELSE
    DISPLAY “VAR-1 NOT = 0”
END-IF.
…
```

When using the COBOL-IT compiler, VAR-1 should first be verified to be NUMERIC before performing the numeric comparison.

As an example:

```
        if var-1 is numeric
          IF VAR-1 = 0
              DISPLAY “VAR-1 = 0”
          ELSE
              DISPLAY “VAR-1 NOT = 0”
          END-IF.
       else
          display "var-1 is not numeric" line 10 col 10
       end-if.
```

3.4 Truncation in DISPLAY of allocated memory

ID: 1145864224

It was not possible to DISPLAY allocated memory that extended beyond the length of the declared linkage variable.

This has been corrected.

3.4 Performance anomaly on INITIALIZE of Variables of USAGE NATIONAL

ID: 1145870456

In some situations the performance of the INITIALIZE and MOVE SPACES TO variables described as USAGE NATIONAL exhibited poor performance.

This has been corrected.

3.4 RECORDING MODE V anomaly

ID: 1145873196

When describing a variable-length record in a file with the RECORDING MODE V clause, in some cases, COBOL-IT would pad the records with spaces, and not create true variable-length records.

This has been corrected.

3.4 Compiler error when variable name starts with the word "copy"

ID: 1145835364

When a variable name started with the word "copy", as in :
01 COPY-AAA PIC 999.

The compiler would return an error as follows:

Error: -AAA: No such file or directory
Error: syntax error, unexpected TOKEN, expecting end of file or COPY or REPLACE or LINKAGE_SECTION

This has been corrected.

3.3 Incorrect length for LINKAGE SECTION item with OCCURS DEPENDING ON

ID: 1145836442

When the length of a variable passed as a parameter to a COBOL program was determined by an OCCURS DEPENDING ON clause in the Linkage Section, it was possible in some cases for the length of the variable to be incorrectly represented as 1 greater than the actual length of the parameter.

This has been corrected.

3.3 Compiling a program with -O -Os causing syntax errors

ID: 1145817864

On a Sun Sparc platform, using the –O and –Os compiler flags could cause compiler errors in certain situations.

This has been corrected.

3.3 Failure of compiler to report error on incorrect INITIALIZE statement

ID: 1145817476

In a case where an INITIALIZE statements was designed to INITIALIZE multiple fields, and where the INITIALIZE statement was incorrectly prematurely terminated, in some cases, the compiler would not report an error.

This has been corrected.

3.3 Error on Numeric Edited field leading sign when DECIMAL-POINT IS COMMA

A data element described as a numeric edited field with a leading sign, and a fixed decimal point, could be incorrectly interpreted when DECIMAL-POINT IS COMMA was declared in SPECIAL-NAMES.
As an example:

```
. . .
    05 data-element  PIC +9(14)V.9(4).
```

This has been corrected.

3.3 Numeric Edited field multiple decimal separators DISPLAYs/PRINT anomaly

ID: 1145807864

When a numeric edited field that contained more than one decimal separator was used in a Screen Section field, the field might be incorrectly displayed after being the target of a MOVE of a numeric

edited literal with fewer decimal separators.

This has been corrected.

3.3 ACCEPT FROM DATE reading COB_CURRENT_DATE

ID: 1145809870

The ACCEPT FROM DATE statement was checking for a value in the environment variable COB_CURRENT_DATE every time it was executed, causing unnecessary overhead that could impact performance.

This has been corrected.

The ACCEPT FROM DATE statement now checks once to determine whether the COB_CURRENT_DATE environment variable is set. If it is set, then the value is stored in cache, and is retrieved by the ACCEPT FROM DATE statement. If it is not set, then COB_CURRENT_DATE is not re-checked, and the ACCEPT FROM DATE statement applies its default behavior of retrieving the date from the system.

3.3 8 fields in a SPLIT KEY cause compiler error

ID: 1145794862

When using VB/ISAM files, it was not possible to use more than 7 fields in a split key.

The limit has be set to 8 fields for a split key.

3.3 LENGTH OF anomaly

ID: 1145793094

Applying the LENGTH OF statement to a variable described as PIC N(n) NATIONAL could produce incorrect results.

This has been corrected.

3.3 Error with VBISAM DELETE

ID: 1145793108

In some situations, when using VBISAM, the DELETE operation could cause the runtime to abort with an attempt to reference unallocated memory (SIGSEGV).

This has been corrected.

3.3 Sort files create memory leak

ID: 1145793632

In some situations, a memory leak was detectable when using the SORT verb.

This has been corrected.

3.3 -falloc-unused-linkage

ID: 1145758674

The –falloc-unused-linkage compiler flag causes the compiler to allocate static memory for level 01 fields in the Linkage Section that are not used in either a USING clause or an ENTRY clause.
If the –falloc-unused linkage compiler flag is not used, and level 01 fields in the Linkage Section are not used in either a USING clause or an ENTRY clause, these fields are initialized to NULL, and no memory is allocated for them.

Note that usage of of a field for which no static memory has been allocated will provoke a Memory Fault. For cases such as described above, where static memory has not been allocated at compile time, it is possible to programmatically allocate static memory to an unused linkage field using the SET [linkage field] to ADDRESS OF [data-pointer] statement, and avoid the Memory Fault condition.

3.3 move-picx-to-pic9:iso

ID: 1145783768

There is a new option “iso” for the compiler configuration flag move-picx-to-pic9.

The compiler configuration flag move-picx-to-pic9 defines the runtime behavior when moving alphanumeric (PIC X(n) USAGE DISPLAY), to non-signed display numeric data items (PIC 9(n) USAGE DISPLAY).

Using the compiler configuration flag move-picx-to-pic9:iso requires that the compiler configuration flag displaynumeric:mf50:yes also be set.

When set to “iso” (when displaynumeric-mf50:yes)

When MOVEing a PIC X field padded with SPACES to a PIC 9 field, the padding SPACES are converted to 0s. As an example, when MOVEing a PIC XX VALUE “2” field to a PIC 99 field, the PIC 99 field would store the target as “02”.

Otherwise (when displaynumeric-mf50:yes)

When MOVEing a PIC X field padded with SPACES to a PIC 9 field, the padding SPACES are not converted. As an example, when MOVEing a PIC XX VALUE “2” field to a PIC 99 field, the PIC 99 field would store the target as “ 2”.

3.3 Improved performance of arithmetic operations

Performance has been improved on arithmetic operations manipulating data items described as USAGE COMP-3 containing more than 20 digits.

3.3 Support for ACCEPT [Screen or Field] TIMEOUT [timeout-value]

The ACCEPT … TIMEOUT phrase allows for a Screen Section ACCEPT statement to be automatically terminated after a number of seconds, as defined in [timeout-value]. If no data has been entered into the ACCEPT statement in the time defined in [timeout-value], then the ACCEPT statement terminates, generating an exception condition, and an exception value.

As an example, with a Screen Section ACCEPT:

```
ACCEPT SCREEN-1 TIMEOUT 5
   on exception
      display "timed out!" line 12 col 10
END-ACCEPT.
```

As an example, with a field-level ACCEPT:

```
ACCEPT element-1 TIMEOUT 5
   on exception
      display "timed out!" line 12 col 10
END-ACCEPT.
```

In these examples, the TIMEOUT is set to 5 seconds.

If any data is entered into the ACCEPT before the designated timeout value, then the timeout timer is restarted.

If no data is entered into the ACCEPT before the designated timeout value, then an exception condition occurs, and the COB-SCR-TIMEOUT exception value of 9001 is generated into the CRT STATUS variable.

Fixes

3.3 Compiler aborting on Screen Section line no variable, and VALUE SPACE

ID: 1145774722

In some cases, when a Screen Section statements that did not explicitly name a variable, and contained a VALUE statement with a figurative constant, the compiler could abort.

This has been corrected.

3.3 MULTIPLY GIVING PIC $$$$,$$9.99

ID: 1145779812

In some cases, using a numeric edited field as the target of a MULTIPLY … GIVING statement could produce incorrect results.

This has been corrected.

3.3 IF SPACE = variable not evaluated correctly

ID: 1145783348

Using a figurative constant as the source field in an IF statement would not produce the expected results. As an example, in the case below, the IF condition did not test as TRUE:

```
77 element-1  PIC X VALUE SPACE.

IF SPACE = element-1
   DISPLAY “element-1 equals SPACE”.
```

This has been corrected. .

3.3 FILLER described as USAGE BIT

When FILLER was described as USAGE BIT, the initialize-filler compiler configuration flag was not being applied.
This has been corrected. In this case, the initialize-filler: [y/n] compiler configuration flag is now applied.

3.3 Data described as USAGE BIT may be given an initial VALUE

ID: 1145773834

Data described as USAGE BIT was not being initialized at program startup, as indicated by the

INITIALIZE FILLER: [y/n] compiler configuration flag, and as indicated by the VALUE clause.

As an example:
```
01 ALL-FLAGS.
  03 FILLER    PIC 1 BIT
   88  ALL-DONE  VALUE B"1"  FALSE B"0".

…
INITIALIZE ALL-FLAGS.
…
```
This has been corrected.

3.3 ACCEPT [Screen] problem when handling fields bigger that 128 characters

ID: 1145745772

When screen input fields were greater than 128 characters, the screen section ACCEPT could limit the input of data to fewer bytes than the screen field size.

This has been corrected.

3.3 IF [procedure-pointer] = NULL statement incorrectly translated to "C"

ID: 1145767946

In certain circumstances, comparing a data item declared as a PROCEDURE-POINTER to NULL in an IF statement could cause incorrect "C" code to be generated.

This has been corrected.

3.3 Default OPTIONAL file behaves different from MF when using OPEN INPUT

ID 1145771864

In Micro Focus when opening a file INPUT that does not have "OPTIONAL" in the SELECT statement causes the program to abort on the open.

In COBOL-IT when the program is compiled with -STD=MF the program does not abort until the first read.

This has been corrected.

3.3 Anomaly debugging source compiled with –fdebug-exec compiler flag

ID: 1145749620

When using the PrintEasy preprocessor, and then debugging while using the –fdebug-exec compiler flag, the debugger could step incorrectly to the wrong line.

This has been corrected.

3.3 Compiler output suppressed whan a copy file is missing

ID: 1145717714
ID: 1145756144

In some cases, compiler output would be suppressed when missing a copy file.

This has been corrected.

3.3 Unfolding doesn't preserve SQL Include Statement

ID: 1145755742
ID: 1145756156
Using the –fkeep-copy-statement allows copyfiles to be preserved in the source, and makes them respond to the fold/unfold commands in the Developer Studio. However, the –fkeep-copy-statement compiler flag did not apply to EXEC SQL INCLUDE statements.

This has been corrected.

3.3 cobc -V produces segment fault if COBOL-IT environment not setup

ID: 1145761234
On a Sun Sparc server, if the cobol-it-setup.sh shell script had not been run, then the 64-bit compiler could return a segment fault error and abort after executing:
>cobc –V

This has been corrected.

3.3 Incorrect variable truncation

ID: 1145756808
In the implicit move of a numeric data item to a numeric edited data item taking place during a *COMPUTE numeric-edited-variable-2 = numeric-variable-1* statement, data could be incorrectly truncated.

This has been corrected.

3.3 Alignment of BINARY SYNC data items

COBOL-IT could place a group item, and the first elementary item underneath it at different offsets, when the elementary items in the group were described as USAGE BINARY SYNC.

As an example, consider the following structure:

```
01 FILLER.
   05 POSRUB-34 PIC S9(04) BINARY SYNC VALUE ZERO.
   05 ELEM-TESTS.
      10 ETEST-ORD PIC S9(8) BINARY SYNC.
      10 ETEST-A PIC S9(4) BINARY SYNC.
```

In this example, the group item ELEM-TESTS was placed at offset 2, while the elementary item ETEST-ORD was placed at offset 4. Correct behavior is to place both the group item and the first elementary item at the same offset.

This has been corrected.

3.3 CBL_ALLOC_MEM

In certain cases, CBL_ALLOC_MEM could cause an infinite loop to be executed in the runtime.

This has been corrected.

3.3 Compiler aborts referencing undefined variable

ID: 1145646746

In certain cases, referencing an undefined variable in the source could cause the compiler to abort.

This has been corrected.

3.3 Screen occurs not working

ID: 1145732636

The use of an OCCURS phrase in a SCREEN SECTION would create an incorrect DISPLAY, in which lines subsequent to the fist would overlay the first line in the DISPLAY.

This has been corrected.

3.3 ACCEPT of ZZZZZ9 screen field not working

ID: 1145732646

On the ACCEPT of a numeric-edited field described as PIC Z(n)9 value 0, the 0 value would be left-justified, when it should have been right-justified.

This has been corrected.

3.3 Using NATIONAL-OF and DISPLAY-OF functions with in special cases

ID: 1145734542

When using reference modification in the target field of a MOVE, where the source data was derived from a NATIONAL-OF or DISPLAY-OF function, undefined results would occur if the first element in the reference modification were described by a variable.

As an example, Formulation 1 below was problematic, whereas Formulation 2 was not:

Formulation 1: MOVE "Hello" TO XXX.
MOVE FUNCTION NATIONAL-OF(XXX 1252) TO UCS-CHAR(POS:10).

Formulation 2: MOVE "Hello" TO XXX.
MOVE FUNCTION NATIONAL-OF(XXX 1252) TO UCS-CHAR(1:10).

This has been corrected.

3.3 Screen Section DISPLAY anomaly

ID: 1145743252

DISPLAYing a Screen from the Screen Section, and then DISPLAYing a line within that same Screen, could cause a DISPLAY anomaly that would cause to the line to be erased from the Screen when the ACCEPT terminated.

This has been corrected.

3.3 COMMIT-ROLLBACK using VBIsam Extfh Driver fixed

The COMMIT and ROLLBACK functions were not fully implemented in the VB/ISAM EXTFH driver.

This has been corrected.

3.3 Incorrect line number reported when error in EXEC … END-EXEC

ID: 1145717644

When a syntax error was reported while using the –fsyntax-only compiler flag, and when that syntax error was inside an EXEC… END-EXEC statement, the line number in the source on which the error occurred was incorrectly reported.

This has been corrected.

3.3 Comments in source block EXEC … END-EXEC syntax checking

ID: 1145717684

In some cases, comments in the source code could prevent syntax checking from being applied to EXEC… END-EXEC statements. This could affect programs using EXEC CICS, EXEC DLI, or EXEC SQL statements.

This has been corrected.

3.3 Compiler output suppressed whan a copy file is missing

ID: 1145717714

ID: 1145756144

In some cases, compiler output would be suppressed when missing a copy file.

This has been corrected.

3.3 Open I-O of optinal VBISAM file returns error 39 if alternate keys declared

ID: 1145728182

If a VBISAM file had one or more alternate keys , and if the file did not exist, and if the file were OPENed I/O, and if optional-file:yes were set in the COBOL configuration file, then the file would not be created, and a file status “39” would be returned.

This has been corrected. The file is now created, and a file status of “00” is returned.

3.3 AUTO not working for ACCEPT

ID: 1145729308

In certain cases, the AUTO attribute did not cause the cursor to automatically advance to the next field when a field was full. In those cases, it was necessary to use the TAB key to pass to the next field.

This has been corrected.

3.3 WRITE ON PRINT FILE WITHOUT AFTER

ID: 1145729768

When a file was OPEN EXTEND, a the first WRITE statement, if it contained no ADVANCING syntax, would behave like a WRITE AFTER 0, when the correct behavior would have been to behave like a WRITE AFTER 1.

This has been corrected.

3.3 Using FILLER in REDEFINES aborts compiler while building the C source

ID: 1145730976

Using FILLER to redefine a variable could cause the compiler to abort when translating COBOL to "C". As an example:

```
05 SSN                      PIC X(09).
05 FILLER REDEFINES SSN.
  07 SSN-3                  PIC X(03).
  07 SSN-2                  PIC X(02).
  07 SSN-4                  PIC X(04).
```

This has been corrected.

3.2

3.2.10

New (E)

3.2 move-picx-to-pic9:raw

There is a new option “raw” for the compiler configuration flag move-picx-to-pic9.

The compiler configuration flag move-picx-to-pic9 defines the runtime behavior when moving alphanumeric (PIC X(n) USAGE DISPLAY), to non-signed display numeric data items (PIC 9(n) USAGE DISPLAY).

When set to “raw”

If the MOVE from PIC X TO PIC 9 field is between fields of the same size, and if spzero is no, then the data is copied with no conversion and no validation.

3.2 COB_CALL_CASE=xul [where x=exact match, u=uppercase, l=lowercase]

3.2 COB_LOAD_CASE=xul [where x= exact match, u=uppercase, l=lowercase]

COB_CALL_CASE and COB_LOAD_CASE should be used together, and affect the behavior of the CALL statement, as regards algorithms applied when locating the target of the CALL statement.

When COBOL-IT executes a CALL “[Symbol]” statement, the default behavior for the runtime is: to look first in memory for:

1- an exact case match of “[Symbol]”,
2- “[Symbol]” in upper case,
3- “[Symbol]” in lower case.

If “[Symbol]” is not found in memory, the runtime will then look for a filename with:

1- an exact case match of “[Symbol]”,
2- “[Symbol]” in upper case,
3- “[Symbol]” in lower case.

The first match is used.

COB_CALL_CASE and COB_LOAD_CASE provide the user with control over this lookup process, as follows:

COB_CALL_CASE=xul controls the behavior of “[Symbol]” lookup in memory.
COB_LOAD_CASE=xul controls the behavior of “[Symbol]” lookup in a filename on disk.

Where the letters xul represent three letters, each of which may be set toY or N, to set whether or not the x (exact match check), u (uppercase check) or l (lower-case check) is performed.

Default settings are :
COB_CALL_CASE=YYY
COB_LOAD_CASE=YYY

An example of a usage, in which only exact case matching would be applied for a symbol in

memory would be, while upper-case and lower-case matching for the filename is preserved:

export COB_CALL_CASE=YNN
export COB_LOAD_CASE=YYY

3.2 -fprotect-linkage

The compiler flag –fprotect-linkage causes the compiler to generate code at the entry point of a program containing a USING xxx clause. This allows for the passing o f parameters that are NULL pointers. In these cases, where NULL pointers are passed, the compiler creates a „fake“ field of the same definition in WORKING-STORATGE, and substitutes it as a reference for the parameter. Doing this will avoid a SIGVEC error if NULL pointers passed through linkage are targets of a READ or WRITE statement.

3.2 Runtime uses TMPDIR setting for storage of temporary SORT files

The internal COBOL SORT now checks for the setting of TMPDIR as the location for the storage of temporary SORT files. If TMPDIR is not set, then the previous behavior is used, which is OS-Dependent. In some operating systems, this can limit the size of files that can be SORTed by the internal COBOL SORT.

Fixes

3.2 Internal 32-BIT COBOL SORT could not sort file > 2GB

When using the 32-bit compiler, files of greater than 2GB could not be sorted. This was because the 32-bit version of the compiler did not use the 64-bit API for temporary files.
This has been corrected.

3.2 Error comparing PIC 9(n) USAGE COMP-6 to a PIC X(n) USAGE DISPLAY

It was possible to get incorrect results when comparining a data item declared as PIC 9(n) USAGE COMP-6 with a data item declared as PIC X(n) USAGE DISPLAY.

This has been corrected.

3.2 Passing a PIC 9 parameter BY VALUE.

ID: 1145706686

3.2 Passing a PIC X parameter BY VALUE.

ID: 1145700370

A CALL [subprogram] USING BY VALUE [parameter] statement could produce incorrect results, when [parameter] was declared as PIC 9 or PIC X, or as a group item containing a single data item declared as PIC 9.

3.2 I/O performances improvements

Performance improvements have been realized in the READ/WRITE operations on SEQUENTIAL and LINE SEQUENTIAL files. The performance improvements can vary between systems, but will be more noticeable in the READ and WRITE operations of smaller records.

3.2 Level 78 did not allow use of figurative constant SPACES in VALUE claues

ID: 1145690088

The compiler could generate an error when compiling a level-78 data item described with VALUE SPACES.

This has been corrected.

3.2 Constants defined with $set/-constant used in PICTURE/LEVEL 78 clauses

A constant defined with the –constant compiler flag, or with the $SET directive can now be used in PICTURE clauses, and as the target of a level-78 VALUE clause.

As an example:

```
$SET myconstant =  10
…
01 my-variable PIC X (myconstant).
78 my-level78    VALUE myconstant.
```

User constant defined with -constant or with $set is now usable in PICTURE clause (and other clause) like Level 78 defined constant

3.2 Write AFTER ADVANCING LINE leaves trailing linefeed character at EOF

ID: 1145688700

The WRITE [record] AFTER ADVANCING LINE statement on a line sequential file could leave a trailing line feed character at the end of the file.

This has been corrected.

3.2 WRITE AFTER ADVANCING PAGE fails to write trailing carriage-return at EOL

ID: 1145695996

The WRITE [record] AFTER ADVANCING PAGE statement on a line sequential file could incorrectly fail to write a a trailing carriage-return at the end of the line.

This has been corrected.

3.2 FUNCTION REM not working properly

ID: 1145679876

The intrinsic function REM could return incorrect results.

This has been corrected.

3.2 Using environment variable in filename fails in CBL_OPEN_FILE Calls

ID: 1145683668

The use of environment variables to establish the location of a file did not work with the CBL_ library routines that are oriented towards file handling, (for example, CBL_OPEN_FILE).

Thus, in the following statement, if the variable “Tape-Name” were declared as follows, the value of the environment variable MYPATH should be prepended to the /workfile.txt to locate the file:

```
01 Tape-Name pic x(40) value "$MYPATH/workfile.txt".
…..
      call 'CBL_OPEN_FILE'
      using
         Tape-Name
         Tape-Access-Mode
         Tape-Deny-Mode
         Tape-Device
```

Tape-Handle
.

This has been corrected.

3.1

3.1.7

New

3.1 Modifications to internal APIs

Version 3.1 includes modifications to internal APIs. Programs compiled with version 3.1 are in-compatible with programs compiled with versions 1.x, 2.x, and 3.0 of the COBOL-IT Compiler Suite Enterprise Edition. When upgrading to version 3.1 from a previous version of the compiler, you must recompile all programs in your system.

3.1 -ftrap-unhandled-exception

The –ftrap-unhandled-exception compiler flag is useful in cases where certain EC- compiler configuration file flags are set to yes, yet ON EXCEPTION/ON SIZE ERROR/ON OVERFLOW language is not present in the COBOL program. In these cases, the information made available to the user is enhanced when the program aborts. As an example, in a case where there is a compiler configuration flag setting of :

EC-SIZE:yes

and where a program contains a statement such as :

compute a = b / c

and where this phrase does not contain an ON SIZE ERROR clause, the program would abort in cases where c=0, and a division by 0 were being performed.

When compiling with the –ftrap-unhandled-exception compiler flag, an informational message is generated, as follows:

divtest.cbl:17: libcob: Uncattched exception: EC-SIZE-ZERO-DIVIDE

Note- this applies to the following EC- compiler configuration flags:

EC-IMP-ACCEPT :yes
For Accept exception
EC-IMP-DISPLAY :yes
For Display exception
EC-SIZE : yes
#For Arithmetic exception
EC-OVERFLOW : yes
For String/Unstring exception

3.1 Conditional compilation $if/$else/$end may now be nested

$IF/$ELSE/$END conditional compilation directives may now be nested to 99 levels.

3.1 -fshare-all-autolock

The –fshare-all-autolock compiler flag causes all files contain a SHARE WITH ALL clause in their SELECT statement to be also be declared implicitly as LOCK MODE IS AUTOMATIC.

3.1 -fshare-all-manulock

The –fshare-all-manulock compiler flag causes all files contain a SHARE WITH ALL clause in their SELECT statement to be also be declared implicitly as LOCK MODE IS MANUAL.

3.1 LIKE Statement can be applied to elementary data items level 01 thru 49

The usage of the LIKE Statement was restricted to data items declared at the 01-level.
The LIKE Statement can now be used to describe elementary data items at level 01 through level 49.

3.1 -fcurdir-include (default is on), -fno-curdir-include

The –fcurdir-include compiler flag causes the search for a COPY file to be first performed in the current directory. The COPY search is performed for files with default extensions, and with extensions described with the –ext compiler compiler flag.

The –fno-curdir-include compiler flag causes the search for a COPY file to not search for a COPY file in the current directory, unless that directory is named by a –I compiler flag, or by a COB_COPY_FILE, or COBCPY environment variable.

3.1 Extensions to the –I compiler flag

The –I compiler flag, which allows the user to name directories to be searched for COPY files, has been enhanced in an number of ways:

Syntax now exists for combining the –I and –ext compiler flags. Also, LIBRARY/MAP declarations may now be used in a –I compiler flag declaration. These enhancements are enabled through an expanded syntax, with the –I compiler flag now accepting a string in the format:

-I <path>[,ext1,ext2,...,extn][@<LibName>] or
-I <command-file>

Where:

<path>	is the path that the compiler will search for the COPY file.
ext1,ext2,…extn	are file extensions to be included when performing the search for the COPY file.
@	represents the “@” character, which is placed befor <LibName>
<LibName>	is the Library named in a COPY statement that contains a reference to a Library. For example: COPY <filename> IN/OF LibName.

<command-file> is the name of an existing file. When –I is followed by the name of an existing file, that file is read, and each line is treated as a parameter of the –I compiler flag.

Rules:

The following rules apply to the treatment of file extensions:

If file extensions are specified, then the compiler will limit its COPY file search to files with these extensions inside the specified <path>.

If no file extensions are specified, then standard extensions and those specified with the -ext compiler flag are used for the COPY file search.

File extensions must be named when the COPY <path> is the current directory (“.”).

To specify that files with no extension should be included in the COPY file search, use a single dot "." . For example, the following string would check for files with no extension, and with the .cpy extension:

```
>cobc  -I /opt/mycopys,.,.cpy
```

Note that if your file is declared with an extension, for example:

```
COPY test.cpy
```

Then, for the purposes of the COPY file search, it is not necessary to apply any extensions. In this case, you would want to check for files with no extension, as follows:

```
>cobc  -I /opt/mycopys,.
```

That is- you would follow <path> by a comma, and then a single dot “.”, indicating that files with no extension should be included in the COPY file search.

The following rules apply to cases where a COPY <filename> IN/OF LibName statement is being interpreted: Consider the case where source code contains the line:

COPY MyCpy IN/OF LIBA.

Note that the default behavior of the compiler in this case is to look in the current directory for a subdirectory LIBA, and to search that directory for files called MyCpy with all of the standard file extensions. If not found, the compiler would continue the search in all <path> specified by the –I compiler flag, and would ignore the LibraryName.

If no @<LibName> is specified in the –I compiler command, then the default behavior of the compiler is assumed.

If @<LibName> is specified in the –I compiler command, then the default behavior of the compiler is not applied. Instead, the <path> named in the –I compiler command is searched for the COPY file. If the COPY file is not located in the <path>, then the search fails.

Examples:

>cobc -I /opt/copy	looks for COPY files with standard extensions in /opt/copy
>cobc -I /opt/copy,.cpy	looks for COPY files with the .cpy extension in /opt/copy
>cobc -I /opt/copy,.,.cpy	looks for COPY files with no extension or with the .cpy extension in /opt/copy
>cobc -I /opt/copy,.,.cpy@LIBA	looks for COPY files with no extension or with the .cpy extension in /opt/copy when resolving a COPY file described as IN/OF LIBA COPY [filename] IN/OF LIBA
>cobc -I <command-file>	reads <command-file> and interprets each line as a command in the –I command string.

Fixes

3.1 Error compiling CALL statement containing literal strings

In some cases, a CALL statement containing multiple literal strings could be incorrectly interpreted by the compiler.

As an example, when compiling

CALL 'myprog' USING 'ABCDEF' 'ABCD'.

The called program would receive 'ABCDEF' ‘ABCDEF’.

This has been corrected.

3.1 Error sorting files with variable sized records

It was possible to encounter errors when sorting files with variable sized records.

This has been corrected both for COBOL-IT’s internal sort algorithms, and for the EXTSM interface.

3.1 Initializations of USAGE COMP fields with USE-DEFAULTBYTE:YES

When the use-defaultbyte compiler configuration flag was set to yes, fields declared with USAGE COMP and USAGE COMP-3 were not initialized.

This has been corrected.

3.1 Using TYPEDEF with REDEFINES

When a TYPEDEF included a REDEFINES, the computing of the memory address of the REDEFINEd field was done incorrectly.

This has been corrected.

3.1 -E compiler flag output

The -E compiler flag no longer produces a dump of the configuration file. This extended listing is only provided by the –t compiler flag.

3.1 DISPLAY statement corrections

Several fixes have been applied to the DISPLAY statement.

3.1 ADD statement anomaly

The ADD statement could behave incorrectly when one of the fields was declared as PIC 9 USAGE DISPLAY and was holding invalid characters.

This has been corrected.

3.1 Incorrect record locking error reporting in VBISAM

When using VBISAM, the failure of a READ [filename] NEXT LOCK statement would correctly return a file status of 51 if the record were locked. However, a subsequent READ [filename] NEXT LOCK statement would then return an EOF file status of 10.

This has been corrected. Now, the subsequent READ [filename] NEXT LOCK statement returns a file status of 51 if the record is still locked, and returns a file status of 00 if the record is available.

3.1 Incorrect behavior using SHARED WITH READ ONLY in EXTFH interface

When using the SHARED WITH READ ONLY clause in the SELECT phrase in some file systems using the EXTFH interface, behaviors were undefined, as the feature was not fully implemented.

This has been corrected.

Developer Studio

1.5.5 Administrator privileges when performing Developer Studio Updates

If the Developer Studio is installed under either the C:\Program Files or C:\Program Files (x86) folders on your Windows system, and if you are running Windows 7 or Windows Vista, then you must have Administrator privileges in order to perform Developer Studio updates. This is because Windows 7 and Windows Vista consider the C:\Program Files and C:\Program Files (x86) folders to be restricted directories.

1.5.5 Object files created by default in Source directory

When no "-o" compiler flag notation is included in the compiler command, the default behavior of the Developer Studio is now to create object files in the same directory as the source file. Note that in prior versions of the Developer Studio, the default was to create object files in the root directory of the project.

1.5.5 Enhancements to Source folding/unfolding

Several enhancements have been applied to the Source folding/unfolding features.

1.5.5 Handling source files of same name in different directories

When the same source name was present in different directory of a project, the Developer Studio Build process could become confused.

This has been corrected.

1.5.5 Running consecutive remote debugging sessions on SunOS

Running consecutive remote debugging sessions on the SunOS could cause errors to be generated.

This has been corrected.

3.0

3.0.6

New

3.0 Contents of compiler configuration file printed in listing file

ID: 1145532702

When a compilation string designates no configuration file, the settings of the default.conf file are printed to the listing file that is named, or implied when using the –t compiler flag. When a compilation string designates a configuration file, using the –conf=xxx.conf compiler flag, then the settings of the named compiler configuration flag are printed in the listing file that is named or implied when using the –t compiler flag.

3.0 OPEN INPUT REVERSE is now supported

ID: 1145616418

The phrase OPEN INPUT REVERSE is now supported for RECORD SEQUENTIAL files with fixed length records. After performing an OPEN INPUT REVERSE, the READ statements return records in reverse order, beginning with the last record.

Example- In a record sequential file with records of 10 characters, and containing data as follows:

AAAAAAAAAA
BBBBBBBBBB
CCCCCCCCCC

The phrase :

```
open input log-file reversed.
read log-file into ws-log-record.
```

Returns the last record, with values CCCCCCCCCC.

3.0 Modifications to internal APIs

Version 3.0 includes modifications to internal APIs. Programs compiled with version 3.0 are incompatible with programs compiled with versions 1.x and 2.x of the COBOL-IT Compiler Suite Enterprise Edition. When upgrading to version 3.0 from a previous version of the compiler, you must recompile all programs in your system.

3.0 Compiler behaviors with -finclude-main and -x

The way the compiler generates the main symbol (executable program entry point) when compiling with -x has changed.

Previous behavior

Previously, the compiler generated a wrapper C module <module-name>_main.c with the main symbol that called the first module (.o or .cob) given as a parameter at the -x command line.

The C module was compiled into a .o and stored in a static ".a" library. Then this library was appended at link time.

New behavior

When compiling with the compiler flag combination **-c -finclude-main,** the compiler inserts the main symbol into the object of the first COBOL module named on the command line.

When compiling with the compiler flag **–x**, and at least one COBOL module has been compiled, the -finclude-main compiler flag is implied, and no static ".a" library produced.

If no COBOL modules are compiled, the **Previous behavior** is used.

Example :

>cobc -c -finclude-main myprog.cob mysub.cob

You may the link the program with:

```
>cobc -x -flink-only myprog.o mysub.o
```

This will generate a `myprog` executable. No static ".a" library is produced, and the main symbol of myprog.o is used.

Example:

>cobc -x myprog.cob mysub.cob

This will generate a `myprog` executable. No static ".a" library is produced

3.0 mf-compat-parser: [y/n]

mf-compat-parser is now "on" by default. The mf-compat-parser may be disabled using

the –fno-mf-compat-parser compiler flag.

When mf-compat-parser is set to on (the default):

A data element, including level-66 and level-88 data names may have the same name as a paragraph, or section. The compiler no longer generates an error in this situation.

3.0 -fregion0

The compiler flag -fregion0 lets you specify that the module that will always execute in region 0 even if called from another region. When called from another region, the module will switch to region 0 on entry and switch back to the calling region at exit.

3.0 COBLPFORM

The environment variable COBLPFORM is now supported, and is compatible with the usage by Micro Focus COBOL. COBLPFORM provides a way to define channel information (CS0, CS1, ...) From MF Doc:

With the implementation of the environment variable COBLPFORM, COBOL-IT can emulate the printer channels C01 through C12 by line feeds and form feeds. The environment variable COBLPFORM provides a syntax that allows the user to define line numbers on the form, and to associate printer channels with those line numbers.

Syntax:

```
COBLPFORM="n: : : :n: : :n: : :n"
export COBLPFORM
```

Parameters:

n	are digits which specify a line number you require for a printer channel. Printer channels with line number 0, or described with mnemonics S01, S02, or CSP, or that are undefined, are set to line 1, at the beginning of the page.

Example:

To set channel 1 to line 1, and channel 12 to line 60.

```
COBLPFORM="1: : : : : : : : : :60"
export COBLPFORM
```

Fixes

3.0 Anomaly using –fno-optimize-move compiler flag

ID: 1145630722

When compiling with the –fno-optimize-move compiler flag, and when moving a variable to itself, (for example: MOVE a-variable TO a-variable.) it was possible to get incorrect results.

This has been corrected.

3.0 Moving spaces to numeric-edited fields

ID: 1145629122

Using the compiler configuration flags:
displaynumeric-mf50:yes
move-spaces-to-display-numeric:yes

did not provide behavior compatible with Micro Focus behavior, where the numeric-edited fields are allowed to be initialized to spaces.

This has been corrected.

3.0 Erroneous syntax aborts compiler

ID: 1145624772

The compiler could abort when encountering incorrect syntax that described a 77-level data item with an 88-level item, and then followed by elementary items.

This has been corrected.

3.0 Infinite Loop calling citEXTFH from a third party EXTFH driver

It was possible to cause an infinite loop when calling the COMMIT/ROLLBACK function of citEXTFH from a third-party EXTFH driver.

This has been corrected.

3.0 exit-program-forced defaults to YES

The compiler configuration flag exit-program-forced is not set to “Yes” by default.

The setting of the exit-program-forced compiler configuration flag affects the behavior of the EXIT PROGRAM statement.

Possible values: yes, no

Default setting:
exit-program-forced: yes

When set to yes,
The execution of the EXIT PROGRAM statement is forced, creating compatibility with non-ISO-standard behavior used by some COBOL compilers.

When set to no,
The EXIT PROGRAM statement behaves according to ISO standards, which state that the EXIT PROGRAM should be ignored if the currently running program was not CALL’ed by a COBOL main program.

3.0 Screen section with control character not properly handled

ID: 1145616196

The compiler would reject some literal declarations that included special characters. As an example, a literal string which included bytes (in hex) 00 13 00 19 … generated compiler errors.

This has been corrected.

The low-value bytes, which were formerly only allowed in a string that was marked as a hexadecimal string, can now be included in an alphanumeric literal string, as long as they are contained in a literal within quotes.

3.0 CRT STATUS is not 4 characters long error

ID: 1145613784

The compiler would return an error if the CRT STATUS variable was not 4 characters long.
As an example, in the code below, the CRT STATUS variable is named as key-status, and key-status is declared as only 3-bytes in length:
…

```
        special-names.
            crt status is key-status.
```

…

```
       WORKING-STORAGE SECTION.

        01 key-status.
            03 key-status-test.
                05 key-type                 pic x(1).
                05 key-code-1               pic 9(2)  comp-x.
            03 key-code-2                   pic 9(2)  comp-x.

       PROCEDURE DIVISION.
           ….
```

This report has been changed to a Warning.

3.0 CONFIGURATION error when no ENVIRONMENT DIVISION defined

ID: 1145613796

The compiler would generate an error when there was a declaration of a CONFIGURATION SECTION when no ENVIRONMENT DIVISION had been previously declared. The ANSI85 cobol definition states that the ENVIRONMENT DIVISION declaration is optional and this should be supported.

As an example, the program below has a declaration for the CONFIGURATION SECTION, but no declaration for the ENVIRONMENT DIVISION.

```
       PROGRAM-ID. TEST1.
       CONFIGURATION SECTION.
       SPECIAL-NAMES.
       . . .
       PROCEDURE DIVISION.
           . . .
```

This has been corrected.

3.0 CONTROL word caused compilation error in SCREEN SECTION

ID: 1145614660

The CONTROL word in the SCREEN SECTION caused a compilation error:

syntax error, unexpected CONTROL, expecting LEADING or TRAILING

For example, note the use of the CONTROL keyword in the screen section example below:

```
       . . .
       screen section.
       01 scr-disp.
         02 scr-disp-indic.
           03 in01                              pic x(25).
       01 disp-screen-1 FOREGROUND-COLOR 2.
           03 line 21 col 2    pic x(25)
              from backroom-msg
              foreground-color  2
              CONTROL IS in01.
```

```
*
  PROCEDURE DIVISION.
  . . .
```

This has been corrected, and this usage is now supported.

3.0 LINE declaration in SCREEN SECTION required a line number

ID: 1145614972
ID: 1145616240

The compiler required that a LINE declaration in the SCREEN SECTION contain a line number. In other COBOLs, the LINE declaration can be used without a line number, indicating "next line".

As an example, in the code below, the first line in the definition of clear-screen references line 5 explicitly. The line below it contains the word "line" without an explicit reference to a line number. This should be interpreted as "line + 1".: Instead, it was generating a compiler error.

```
. . .
WORKING-STORAGE SECTION.
screen section.
01 clear-screen.
    03 line 5 pic x(80) from spaces.
    03 line pic x(80) from spaces.
    03 line pic x(80) from spaces.
. . .
PROCEDURE DIVISION.
. . .
```

This has been corrected. The LINE declaration can now be used in the screen section without a line number, and this is interpreted as "next line".

3.0 SPECIAL-NAMES entry required a CONFIGURATION SECTION

ID: 1145615038

A SPECIAL NAMES entry required a CONFIGURATION SECTION declaration.
Declaring a SPECIAL NAMES section without previously declaring the CONFIGURATION SECTION caused a compilation error with COBOL-IT:

Error: syntax error, unexpected SPECIAL-NAMES, expecting end of file.

As an example, below, the special-names declaration is not preceded by a declaration of a CONFIGURATION SECTION.:

```
environment division.
special-names.
    crt status is key-status.
data division.
file section.
working-storage section.
```

This has been corrected. The condition no longer generates an error.

3.0 64-bit installation for VS 2010 installs 32-bit setenv_cobolit.bat

ID: 1145591650

In some versions of COBOL-IT, the installation of the 64-bit Windows version of the compiler configured for use with Visual Studio 2010 would cause the 32-bit version of setenv_cobolit.bat to be installed.

This has been corrected.

3.0 Wrong XDD key information when using redefines

ID: 1145598764

In an FD containing a key definition which has multiple REDEFINES, the XDD file generated by compiling with the –fgen-xdd compiler flag could create an incorrect XDD file.

This has been corrected.

3.0 Problems when compiling at the same time

ID 1145600504

When two programs were being compiled at the same time, it was possible for the output files to be misnamed. That is, if a prog1.cbl and a prog2.cbl were being compiled at the same time, it was possible that prog1.cbl would create a prog2.dll, while prog2.cbl created a prog1.dll.

This has been corrected.

3.0 Incorrect results on MOVE to a LINKAGE data item

ID: 1145605862

When using Visual Studio 2008, under some circumstances, a MOVE of an alphanumeric value to an alphanumeric data item in the Linkage Section with a different declaration could produce incorrect results.

This has been corrected.

3.0 Incorrect results on MOVE to a COMP-2 data item

ID: 1145610618

When using Visual Studio 2008, under some circumstances, a MOVE of a signed integer to a field described with USAGE COMP-2 could produce incorrect results.

This has been corrected.

3.0 -sysout=<file> no longer causes redirection of DISPLAY UPON CONSOLE

DISPLAY … UPON CONSOLE statements were being redirected to <file> when using the compiler flag –sysout-<file>. This has been corrected. When using the compiler flag –sysout=<file>, only the DISPLAY … UPON SYSOUT statements are redirected to <file>.

3.0 console-is-sysfile: [y/n]

The compiler configuration flag console-is-sysfile, when set to yes, causes DISPLAY …. UPON CONSOLE statements to be redirected to <file> when the compiler flag –sysout=<file> is used. This allows users to maintain a pre-2.11 behavior, which was determine to be defective.

3.0 RETURNING [field] anomaly

In some conditions, when the target of a RETURNING clause was declared as a character (PIC X) field that was smaller than 4 bytes, a

`CALL "program " returning WS-RETURN-CODE.` Statement did not retrieve the returned value correctly: This has been corrected.

3.0 WRITE AFTER/BEFORE LINE/PAGE/CHANNEL anomaly

In some situations, the phrase WRITE AFTER/BEFORE LINE/PAGE/CHANNEL would add an extra carriage return, and in some situations, it would not add a carriage return. Both of these cases have been corrected.

Developer Studio

1.4.8 Version 1.4.8 installs as update to version 1.4.3

Version 1.4.8 of Developer Studio requires the RSE version 3.3, and COBOL-IT version 3.0

To install a version of Developer Studio version as an update to version 1.4.3:

First, make sure that you have installed COBOL-IT version 3.0, and Developer Studio v 1.4.3.

Use the options provided by Eclipse to install the RSE version 3.3 and the new Developer Studio version as follows:

First, install RSE version 3.3:

Go to menu "Help" -> 'Install New Software' -> "Available Software Site"

In the Available Software Site dialog screen,

Click "Add"

On the Add Site dialog Screen, enter:

Name: "RSE 3.3"
Location: http://download.eclipse.org/tm/updates/3.3

Click OK
to close the Add Site dialog screen, and return to the Available Software Site dialog screen.

On the Available Software Site dialog screen,

Drop down the "Work with" combo-box, and select your newly added RSE 3.3 entry.

Select the TM and RSE 3.3 Main Features and Optional Add-ons.
Click Next to Continue.

Where software is already installed, modifications need only be done. This is handled

automatically by Eclipse.

Read the terms of the Eclipse License Agreement.

Click to Accept the terms.
Click on Finish to begin the software installation.

Re-start Eclipse, as prompted. Note that this is not a restart of Windows, merely of Eclipse. When Eclipse is restarted, it prompts for a Workspace.

Click OK.

Next, install the version of COBOL-IT available through the Update function:

Select Help > Check for Updates from the Main Menu.
Select CobolITEclipse.
Click on Next to continue.

Click to Accept the terms.
Click on Finish to begin the software installation.

Re-start Eclipse, as prompted. Note that this is not a restart of Windows, merely of Eclipse. When Eclipse is restarted, it prompts for a Workspace.

Click OK.

1.4.8 Debugger crashes showing variables from LOCAL-STORAGE SECTION

ID: 1145630392

The Variables View of the Debugger Perspective could crash when showing variables from the LOCAL-STORAGE SECTION.

This has been corrected.

1.4.8 Debugger crashes when removing values in expressions

ID: 1145610618

The Developer Studio could crash when removing a variable from the Expressions View.
For example, if you were to add a variable to the Expressions View, then select this variable by clicking on it, and press the "Del" button, and then hit [enter], you could cause the Developer Studio to abort.

This has been corrected.

1.4.8 Anomaly where variables of a table are shown n expressions view

ID: 1145589794

The Developer Studio did not have a way to describe an element within an indexed group.

For example:
ee-group(INDEX).ee-field

or multi-dimensional tables:
ee-group1(INDEX1).ee-group2(INDEX2). ee-group3(INDEX3)ee-field

This has been corrected.

1.4.8 Windows anomaly: Using ".\" for the list file name causes compile error

ID: 1145598080

In the Source>Properties>COBOL Properties>Language tab, using the notation ".\" (without quotes) in the field titled "Generate file as list file (-t file)" would cause an "invalid argument" compiler error.

This has been corrected.

1.4.8 Can't set breakpoints in a copybook used in PROCEDURE DIVISION

ID: 1145604168

It was not possible to set a breakpoint in a copybook that was used in the PROCEURE DIVISION.

This has been corrected.

1.4.8 Open declaration "F3" doesn't work if the copybook is in a separate directory

ID: 1145604190

Open declaration "F3" doesn't work if the copybook is in a separate directory

1.4.8 Debugger crashes during the start

ID: 1145608400

The debugger could abort at startup if there were a variable described in the Expressons View which was not available in the main COBOL program. As an example, if you called a subprogram, and added a variable to the Expressions View, and then restarted the main program, you could cause the debugger to abort.

This has been corrected.

2.10

2.10.1

New (E)

2.10 -fauto-load-symb

The –fauto-load-symb compiler flag provides additional control, as regards static symbol definition. When the -fauto-load-symb compiler flag is used, the compiler load the compiler configuration file static.symb and user.symb.

static.symb is provided with the compiler distribution and should not be changed by the user (it is overwritten when the compiler is updated. Currently it include symbol declared by Pro*Cob and Tuxedo.

user.symb is user-definable and may be placed in the config directory of the COBOL-IT installation, or in the current directory. If user.symb is missing, no error is generated.

Both files may include static-link declarations.

2.10 Initialize-pointer:yes/no

Default : yes

When set to no, elementary fields described as USAGE INDEX, USAGE POINTER, USAGE PROGRAM-POINTER are not initialized by the INITIALIZE verb.

2.10 Support for a CALL prototype

See **Sample Code** below.

General Rules for the CALL prototype

- ENTRY declarations must use parameters declared either as TYPEDEF, or with the reserved word ANY.
- The external program must be properly structured. That is, it must contain a declaration for each of the IDENTIFICATION DIVISION, ENVIRONMENT DIVISION, DATA DIVISION, and PROCEDURE DIVISION.
- The DELIMITED clause is supported in the prototype definition, for example:

```
entry
            C-FUNCTION-VAL
            c-call using
                by reference  schar delimited
```

When the DELIMITED clause is used, strings passed to the “C” function are automatically null-terminated.

Sample Code

prog.cbl

```
COPY "cproto.cpy".
IDENTIFICATION   DIVISION.
PROGRAM-ID.      prog.
ENVIRONMENT      DIVISION.
CONFIGURATION SECTION.
DATA DIVISION.
WORKING-STORAGE SECTION.

01 grpstr.
   03 AV-STR PIC XX VALUE 'AV'.
   03 STR PIC X(10) VALUE "STR10".
   03 AP-STR PIC XX VALUE 'AP'.
   03 zz-STR PIC 0  VALUE X'00'.
01 A1  PIC 99 VALUE 1.
01 A2  PIC 99 VALUE 2.
01 A3  PIC 99 VALUE 3.
01 A4 PIC 99 VALUE 4.

PROCEDURE        DIVISION.

CALL "cfunction_val" USING "vallit" 1 2 3 4.
CALL "cfunction_ref" USING "reflit" 1 2 3 4.
DISPLAY "'" grpstr "'"
CALL "cfunction_val" USING STR A1 A2 A3 A4.
CALL "cfunction_ref" USING STR A1 A2 A3 A4.
DISPLAY "'" grpstr "'"
CALL "cfunction_any" USING grpstr function BYTE-LENGTH(grpstr).

.
```

cproto.cpy

```
 program-id. "c_typedefs" is external.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  schar                   pic x is typedef.
 77  uns-schar               pic  9(2)  comp-5 is typedef.
 77  short                  pic s9(4)  comp-5 is typedef.
 77  uns-short              pic  9(4)  comp-5 is typedef.
 77  int                    pic s9(9)  comp-5 is typedef.
 77  uns-int                pic  9(9)  comp-5 is typedef.
 77  long                   pic s9(9)  comp-5 is typedef.
 77  uns-long               pic  9(9)  comp-5 is typedef.
 77  l-long                 pic s9(18) comp-5 is typedef.
 77  uns-l-long             pic  9(18) comp-5 is typedef.
 end program "c_typedefs".

 program-id. "c_typedefs" is external.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
special-names.
     call-convention 3 is Pascal
     call-convention 0 is c-call.
$set constant C-FUNCTION-VAL "cfunction_val"
$set constant C-FUNCTION-REF "cfunction_ref"
$set constant C-FUNCTION-ANY "cfunction_any"
```

```
       PROCEDURE DIVISION.
        entry
           C-FUNCTION-VAL
           c-call using
               by reference schar delimited
               by value  uns-schar
               by value uns-short
               by value  uns-int
               by value uns-l-long
           returning         int

        entry
           C-FUNCTION-REF
           c-call using
               by reference schar delimited
               by reference  uns-schar
               by reference uns-short
               by reference  uns-int
               by reference uns-l-long
           returning         int

        entry
           C-FUNCTION-ANY
           c-call using
               by reference  any
               by value      uns-int
           returning         int
        .
       end program "c_typedefs".
```

cfunc.c

```
#include "stdio.h"

int cfunction_val(char *s, char c1, short c2, int c3, long long
c4)
{
    printf("'%s' %d %d %d %d %lld\n", s, strlen(s), (int)c1,
(int)c2, c3, c4);
}

int cfunction_ref(char *s, char *c1, short *c2, int *c3, long long
*c4)
{
    printf("'%s' %d ", s, strlen(s));
    if (c1) {
        printf("%d ",  (int)(*c1));
    } else {
        printf("<NULL>");
    }
    if (c2) {
        printf("%d ",  (int)(*c2));
```

```
    } else {
        printf("<NULL>");
    }
    if (c3) {
        printf("%d ", (int)(*c3));
    } else {
        printf("<NULL>");
    }
    if (c4) {
        printf("%lld\n",  (*c4));
    } else {
        printf("<NULL>\n");
    }
}

int cfunction_any(void *any , int c1)
{
    printf("'%s' %d\n", (long long)any, (int)c1);
}
```

2.10 Multiple Sections may have the same name

When using the –fmf-compat-parser compiler flag, the compiler allows multiple SECTIONs in a program to have the same name. Note however that Paragraph Names within a SECTION must be unique to the SECTION.

2.10 A single listing file may be created for multiple programs compiled with -b

When using the –b compiler flag to combine several programs into a single library, and using the –t [filename] compiler flag, a single listing is created that includes all of the programs in the library.

2.10 Internal handling of constants created with -constant compiler flag

The COBOL-IT compiler has modified the manner in which it handles constants created through the use of the $CONSTANT directive, or with the use of the *–constant=[value]* compiler flag, to achieve a higher level of compatibility with other COBOL implementations.

2.10 Berkeley DB interface no longer requires DB_HOME or XA mode be set

In the Berkeley DB Interface, DBDEXTFH has been enhanced so that it no longer requires DB_HOME to be set, or XA mode to be activated.

2.9

2.9.12

New

2.9 Better control of name mapping for static and external link

When the target of a CALL statement is resolved with the use of the static-link: or external-link: compiler configuration flag, the name comparison is not dependent on case. Consider a case where the COBOL program contains the code:

CALL “program1”.

And the compiler configuration file contains the entry:

static-link: PROGRAM1

With this enhancement, the lower-case program1 is matched to the upper-case PROGRAM1. Note that the programs named in static-link: and external-link: entries in the compiler configuration file should always be entered in upper-case.

2.9 cobcrun -V display version and license information

The command cobcrun –V now displays information about the version of COBOL-IT, and the Subscription.

2.9 Debugger support of USAGE BIT fields

The Developer Studio Integrated Debugger now displays, and allows the user to change the value of a field described as USAGE BIT.

2.9 Preliminary support for PIC G USAGE DISPLAY-1

PIC G is an alias of PIC N, and USAGE DISPLAY-1 is an alias of USAGE NATIONAL.

Double-byte strings are supported in constants and variables. For example:

78 dbl-byte-string1 VALUE G”*dbl-btye-encoded-string*”.

78 dbl-byte-string2 VALUE GX”*hex dbl-btye-encoded-string*”.

77 dbl-byte-var1 PIC G VALUE G"*dbl-btye-encoded-string*".

77 dbl-byte-var2 PIC G VALUE GX"*hex dbl-btye-encoded-string*".

77 dbl-byte-var3 PIC G USAGE DISPLAY-1.

2.9 COB_CURRENT_DATE

The COB_CURRENT_DATE runtime environment variable can be used to control the results of the ACCEPT FROM DATE YYYYMMDD statement. COB_CURRENT_DATE holds a date in the format “yyyy-mm-dd”.

Usage:

In Windows:
>set COB_CURRENT_DATE=2011-01-31
….
In Linux/Unix:
>export COB_CURRENT_DATE=2011-01-31

77 the-date PIC 9(8).

….
ACCEPT the-date FROM DATE YYYYMMDD .
DISPLAY the-date line 10 col 10.

Alternatively, you can use the SET verb to set COB_CURRENT_DATE, as follows:

SET ENVIRONMENT "COB_CURRENT_DATE" TO "2011-01-31".

In each of these cases the variable the-date would be returned as 20110131 instead of the current date, as would be expected if COB_CURRENT_DATE were not set.

2.9 Support for " Exhibit Named" syntax

ID: 1145555284

The EXHIBIT NAMED statement is now supported.

The EXHIBIT NAMED statement provides a way to extract the value of a named variable and write it to an error file.

Example:

```
MOVE 10 TO MYNUM.
MOVE "HELLO WORLD" TO MYCHAR.
EXHIBIT NAMED MYCHAR.
EXHIBIT NAMED MYNUM.
```

Output:
MYCHAR = HELLO WORLD
MYNUM = 10

2.9 Internal handling of Interrupts in Windows

In Windows , the COBOL-IT Compiler for Visual C 2008 and Visual C 2010 has changed the way it interrupts and communicates with an application. RPC is now used to enable "attach" and debug functions in programs that have no console-like Services.

2.9 -E compiler flag includes path information on copybook

ID : 1145531342

Using the -E compiler flag, the references to expanded copy files now include path information on the commented-out reference to the copy file. As an example, in the sample below, the program was compiled with –I .\copy –ext sl . Note that the path to the SELECT statement is preserved:

```
*"./copy/reswords.sl"
*
    SELECT reswords ASSIGN TO "reswords"
    ORGANIZATION IS INDEXED
    ACCESS IS DYNAMIC
    RECORD KEY IS reserved-word
    FILE STATUS IS reswords-stat.
```

2.9 Cobfunc( ) and cobcancel( ) don't cancel variables changed in COBOL

ID : 1145531950

cobcancel( ), called from a "C" program to cancel a COBOL-IT program resident in memory,

required that the program name, and program-id be identical, and have the same case, or the target would not be resolved. Now, either the program-name or program-id can be used, and there is not case sensitivity in order to resolve the call.

2.9 -makesyn « old-value=new-value »

ID : 1145537160

The -makesyn "oldvalue=newvalue" compiler flag provides compatibility with the MAKESYN directive.

A COBOL verb or field-name may be used as “old-value”. The COBOL-IT compiler will replace all instances of this “old-value” with the “new-value” when compiling.

CAUTION- While this provides an equivalent capability to the implementation of the MAKESYN directive in other COBOLs, the order of the parameters is reversed. COBOL-IT requires that the “old-value” be listed first, and followed by the “new-value”.

2.9 -sysout=filename [,S/L [,Min [,Max]]] provides OUTDD compatibility

ID : 1145532646

ID: 1145507752

The cobc –sysout=output-file compiler flag now appends output to an existing output-file instead of removing the existing output-file, and creating a new one.

When using the –sysout=filename compiler flag, it is now possible to describe whether the output should be written in binary sequential, or line sequential format. When the line sequential format is indicated, minimum, and maximum lengths may also be indicated. After a maximum length, a line feed (or CR/LF in Windows) is inserted, and the output string is continued.

General format

-sysout=filename [,S/L [,Min [,Max]]]

Where:

- , may be replace by “:”
- No spaces are allowed in the compiler command
- S = Sequential (Binary Sequential)
- L = Line Sequential
- If a Max value is not given for Sequential files, Max = Min.
- If a Max value is not given for Line Sequential files, Max = Min, and Min = 0.

The runtime will write as many records as needed. If the data written is smaller than the Min value, then the runtime will pad the output with SPACES up to min-length in the line-sequential output file, and will pad the output with LOW-VALUES up to min-length in the sequential (binary sequential) output file.

2.9 Compiler error when FD record size mismatch is detected

ID: 1145524976

COBOL-IT generated a compile error if the RECORD CONTAINS phrase in the FD indicated a record size that does not match the actual length of the record. The default behavior of the compiler is now to ignore this.

It is possible to enable this compiler error check by setting
mf-compat-parser:n

in the compiler configuration file.

2.9 mf-compat-parser :[y/n]

The mf-compat-parser compiler configuration file flag causes the COBOL-IT to match certain Micro Focus behaviors. These include :

- Parsing of line continuation characters
- Relaxed syntax check on RECORD CONTAINS phrase in the FD

The default is set to « y ».

To disable these compatibilities, set mf-compat-parser :n in the compiler configuration file.

2.9 LENGTH-AN( ) intrinsic function

ID : 1145494758

The LENGTH-AN intrinsic function returns the length of an argument, in number of alphanumeric character position. It may be applied to an alphanumeric, national, or numeric data item, or literal.

Example :

```
MOVE FUNCTION LENGTH-AN(CONTROL-FLAG) TO VAR1.
```

2.9 -fas400-like (or in configuration file "as400-like:yes")

2.9 as400-like :yes

The –fas400-like compiler flag causes field declared with the LIKE clause to be described as a PIC X (other field's byte size).

This may also be accomplished by adding

as400-like :yes

to the compiler configuration file.

2.9 SELECT... WITH ROLLBACK is now supported

ID: 1145498994

In the LOCK MODE Clause of the SELECT phrase, the WITH ROLLBACK phrase is now supported by COBOL-IT.

General Format

```
[LOCK MODE IS      {MANUAL    WITH LOCK ON [MULTIPLE] {RECORD }} ]
                                                    {RECORDS}

                   {AUTOMATIC WITH LOCK ON [MULTIPLE] {RECORD }}
                                                    {RECORDS}

                   {EXCLUSIVE                                  }

                   [ WITH ROLLBACK ]
```

General Rules

- The file system must support COMMIT and ROLLBACK, otherwise the WITH ROLLBACK clause, COMMIT statement and ROLLBACK statement have no effect.
- When the WITH ROLLBACK phrase is used in the LOCK MODE clause, a file may use the COMMIT and ROLLBACK statements.
- WITH ROLLBACK implies LOCK MODE IS AUTOMATIC, as any of the I-O statements WRITE, REWRITE, DELETE automatically place a lock on the record that is the target of the I-O statement. These locks are released when either the COMMIT or ROLLBACK statement is executed.
- The effect of the ROLLBACK statement is that all record updates, through WRITE, REWRITE, DELETE statements since the last COMMIT or ROLLBACK statement are nullified, or "rolled back".
- The effect of the COMMT statement is that all record updates, through WRITE, REWRITE, DELETE statements are flushed to from buffers to the data source, and locks released.

2.9 The COBOL-IT Region Interface

For the purposes of this documentation, there are 4 sample programs, and a compile script which are designed to highlight key information about the COBOL Region Interface.

These sample programs are:

comp.sh	The compile script
testregion.c	The main program
proga.cob	COBOL program CALL'ed by testregion.c
progb.cob	COBOL program CALL'ed by proga.
progfail.cob	

The compile script used, and the sample programs are all included at the end of this chapter.

Compiling and running the demo

1- Run the compile script-

comp.sh

```
cobc -fthread-safe -m proga.cob
cobc -fthread-safe -m progb.cob
cobc -fthread-safe -m progfail.cob
cobc -x testregion.c
```

Please note that all of the programs used in the demo are compiled with the –fthread-safe compiler flag. It is a requirement that all programs that are involved with regions MUST be compiled the the –fthread-safe compiler option. For more information on the –fthread-safe compiler flag, please reference "Guidelines for thread-safe programs" in "Getting Started with COBOL-IT".

To enable debugging of the "C" code, the compiler flags –G –save-temps should be added to each of the compile commands in comp.sh.

2- After compiling the programs, run the testregion executable:

>testregion

Testregion calls a COBOL program called “proga”, which calls a subprogram “progb”.

The demo shows :

- How to initialize a region, and call a program in that region
- How to cancel a program
- How to use cob_set_exit_rtd_proc with setjmp/longjmp

The REGION API:

All API functions get as a parameter the runtime data from region 0 (the main program region). The runtime data from region 0 is retrieved with a call to cob_get_rtd( ) at program startup. Note that “rtd” is an abbreviation of “runtime data”.

The API functions:

```
unsigned int cob_enterprise_get_current_region (cit_runtime_t * r0rtd);
```

Returns the current region number.

Input :

cit_runtime_t * r0rtd	Region 0 (main region) rtd

```
cit_runtime_t * cob_enterprise_set_current_region (cit_runtime_t * r0rtd,
unsigned int region);
```

Returns an rtd of the region number 'region'. The region is initialized if needed.

Input :

cit_runtime_t * r0rtd	Region 0 (main region) rtd
unsigned int region	Requested region, number between 0 and 15

```
void cob_enterprise_cancel_region(COB_RTD, unsigned int region, int full_cancel)
;
```

Cancels all programs started in a region

Input :

cit_runtime_t * r0rtd	Region 0 (main region) rtd
unsigned int region	Region number to cancel between 0 and 15
int full_cancel	If not 0, the programs are unloaded from memory and memory allocated is freed.

How to call a progam (and all sub programs) in a region

When a region is set to be the current active region, it returns an rtd that must be used as the first parameter for all calls to the runtime library.

To call a program in a region :

- Set the current active region
- Use the returned rtd to resolve and call the program

For example:

```
union {
int (*func)();
void *func_void;
} unifunc;
cit_runtime_t * rtd_region;

rtd_region = cob_enterprise_set_current_region(R0_rtd, region);
unifunc.func_void = cob_resolve (rtd_region, "proga");
if ( unifunc.func_void == NULL ) {
cob_call_error (rtd_region);
}
unifunc.func (NULL,NULL,NULL);
```

WARNING: As a general rule, you should never call a runtime function giving as a first parameter an rtd that does not reference runtime data of the current active region.

The sole exception to this rule would be a call to cob_enterprise_cancel_region.

SetJump/LongJump

In the Region 0 the C program may call 'cob_set_exit_rtd_proc' to set up a C function that will be called if for any reason the runtime aborts. Note that this may also be called if the COBOL program executes a STOP RUN.

Used with setjmp/longjmp this provides a way to recover from an error.

The sample program below shows how to recover from an error which is caused by CALL'ing a subprogram that does not exist:

void cob_set_exit_rtd_proc (cit_runtime_t * r0rtd, cob_rtd_exit_proc cob_exit_proc);

Input :

cit_runtime_t * r0rtd	Region 0 (main region) rtd
cob_rtd_exit_proc	Followed by the pointer to the exit proc
cob_exit_proc	A pointer to a C function getting as its first parameter the rtd of the current region and as its second parameter an int representing the exit status.

comp.sh

```
cobc -fthread-safe -m proga.cob
```

```
cobc -fthread-safe -m progb.cob
cobc -fthread-safe -m progfail.cob
cobc -fthread-safe -x testregion.c
```

testregion.c

```
#include <stdio.h>
#include <assert.h>
#include "libcob.h"

jmp_buf jump_buffer;

void cob_exit_proc (COB_RTD, int status)
{
    /* terminate the current region */
    cob_terminate_exec(rtd );

    /* then go back to call*/
    longjmp(jump_buffer, 1);
}

void call_prog (COB_RTD, int region, char *prog_name, char *
mess) {
    union {
        int (*func)();
        void    *func_void;
    } unifunc;
    cit_runtime_t * rtd_region;
    rtd_region = cob_enterprise_set_current_region(rtd, region);
    unifunc.func_void = cob_resolve (rtd_region, prog_name);
    if ( unifunc.func_void == NULL ) {
        cob_call_error (rtd_region);
    }
    printf("\nregion %s\n", mess);
    unifunc.func (NULL,NULL,NULL);
}



int main (int argc, char **argv)
{
    COB_RTD = cob_get_rtd();
    cit_runtime_t * rtd_region1;
    cit_runtime_t * rtd_region2;

    cob_init (rtd, 0, NULL);

    /* set up termanate Proc in Region 0*/
    cob_set_exit_rtd_proc(rtd,cob_exit_proc);

    /* test Long jump*/
```

```
    if (setjmp(jump_buffer) == 0) {

        call_prog(rtd, 0, "proga", "main");
        call_prog(rtd, 0, "proga", "main");

        call_prog(rtd, 1, "proga", "r1");
        call_prog(rtd, 1, "proga", "r1");
        call_prog(rtd, 2, "proga", "r2");
        printf("\n cancel region 1\n");
        cob_enterprise_cancel_region(rtd, 1, 0);
        call_prog(rtd, 0, "proga", "main");
        call_prog(rtd, 1, "proga", "r1");
        call_prog(rtd, 2, "proga", "r2");
        cob_enterprise_cancel_region(rtd, 2, 0);
        call_prog(rtd, 2, "proga", "r2");

        call_prog(rtd, 2,  "progfail",  "r1");
        printf("We should never display this message");

    } else {
        printf("After long jump");
        call_prog(rtd, 1, "proga", "r1");
    }



}
```

proga.cob

```
       IDENTIFICATION   DIVISION.
       PROGRAM-ID.      proga.
       ENVIRONMENT      DIVISION.
       CONFIGURATION SECTION.
       WORKING-STORAGE SECTION.
       01   fa PIC X(30) VALUE "INITITAL VALUE".

       PROCEDURE        DIVISION.
       DISPLAY "PROGRAM A " fa  .
       MOVE "New data" TO fa.
       CALL "progb"  .
       CALL "progb"  .
       DISPLAY "PROGRAM A CANCEL ProgB" .
       CANCEL "progb".

       CALL "progb"  .
```

progb.cob

```
       IDENTIFICATION   DIVISION.
       PROGRAM-ID.      progb.
       ENVIRONMENT      DIVISION.
```

```
CONFIGURATION SECTION.
WORKING-STORAGE SECTION.
01   fa PIC X(30) VALUE "INITITAL VALUE".

PROCEDURE        DIVISION.
DISPLAY "PROGRAM B " fa .
MOVE "New data" TO fa.
```

progfail.cob

```
IDENTIFICATION   DIVISION.
PROGRAM-ID.      progfail.
ENVIRONMENT      DIVISION.
CONFIGURATION SECTION.
WORKING-STORAGE SECTION.
01   fa PIC X(30) VALUE "INITITAL VALUE".

PROCEDURE        DIVISION.
DISPLAY "PROGRAM Fail " fa  .
MOVE "New data" TO fa.
CALL "FAIL"  .
```

Fixes

2.9 Developer Studio Remote debugger aborts

ID: 1145557690

When operating in a remote project, and auto-refreshing a large table in the "Variable" view, the Developer Studio would crash is the table were too large.

This has been corrected.

2.9 EXTSM Result incorrect

ID: 1145551864

Using an External Sort module on a line sequential file with multiple keys, incorrect results were returned.

This has been corrected.

2.9 XDD information recorded incorrectly for comp, comp-4, comp-x

ID: 1145551520

The –fgen-xdd compiler flag was generating XDD files with the wrong field-type information for fields described as USAGE COMP, USAGE COMP-4, and USAGE COMP-X. These fields were

incorrectly being mapped to CompUnsigned.

They are now correctly being mapped to BinaryUnsigned.

2.9 Move-picx-to-pic9-mf50 anomaly fixed

ID: 1145555218
ID : 1145547952
ID : 1145541246
ID: 1145494832

In a case where the compiler configuration flag "move-picx-to-pic9:mf50" was set, and where an alphanumeric data item containing a numeric value was MOVE'd to a numeric data item, the result of the MOVE did not match the MF behavior.

These have been corrected.

2.9 Initialize of SIGNED field produced incorrect results

ID : 1145545266

It was possible for an INITIALIZE of a variable that was declared as SIGN LEADING SEPARATE or SIGN TRAILING SEPARATE to cause the compiler to abort.

This has been corrected.

2.9 Line-continuation character compatibility tests resolved

ID : 1145545950
ID : 1145541268
ID : 1145542044
ID: 1145443876
ID: 1145522380

Several tests were run with variables including line continuation characters, with behaviors mapped against the Micro Focus compiler.

Differences in behavior were resolved.

2.9 Key information incorrectly stored in xdd

ID : 1145546404

In some circumstances the –fgen-xdd compiler flag would incorrectly represent the key in the XDD that was produced. When this occurred, the c-tree SQL engine could not use the index.

This has been corrected.

2.9 Problem in IF statement with variable declared USAGE BIT

ID : 1145548706
ID : 1145548730

Referencing a variable described with USAGE BIT in an IF statement could compiler errors.

These have been corrected.

2.9 The phrase « b-not » was not supported

ID : 1145548724

b-not was flagged as a reserved word, but not supported.

This has been corrected.

2.9 Preprocess compiler flag could not be applied in a remote compile

ID : 1145550582

In the Developer Studio, the –preprocess compiler flag caused the compiler to abort.

This has been corrected.

2.9 Location of DEPENDING ON clause in table

ID : 1145539912

Customer requires the DEPENDING ON clause to come directly after the variable OCCURS clause in a table, as follows:

77 NUM-TIMES PIC 99 VALUE 20.

05 STAR-TABLE OCCURS 1 to 20 TIMES
ASCENDING KEY IS A-LAST-NAME INDEXED BY INDEX-1.
DEPENDING ON NUM-TIMES....

This has been corrected.

2.9 COPY statement produces a compilation error

ID : 1145541288

The compiler could interpret a period « . » that was part of a copyfile name which was not surrounded by quotes as an end-of-sentence.

This has been corrected.

2.9 Conditional directive anomalies

ID : 1145536382
ID : 1145509794
ID: 1145525100

If a program used the conditional $SET to set a condition, and then used the $IF conditional compillation directive to test if the condition was set, the $IF statement would not detect that the condition was set.

This has been corrected.

2.9 Default-byte :0 causes ADD anomaly

ID : 1145537374

Using the compiler configuration flag default-byte :0 could cause the ADD operation to produce incorrect results in some cases.

This has been corrected.

2.9 COBITOPT environment variable not showing in listing

ID : 1145509262

When using the –t compiler flag to generate a listing, the COBITOPT environment variable was not listed.

This has been corrected.

2.9 Compile error if IDENTIFICATION DIVISION declaration is missing

ID: 1145496908

COBOL-IT no longer requires the IDENTIFICATION DIVISION declaration.

As an example, the following code, which is supported by other COBOL compilers, is now supported by COBOL-IT:

```
PROGRAM-ID. TESTIT IS INITIAL PROGRAM.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
PROCEDURE DIVISION.
    DISPLAY "HELLO WORLD".
    STOP RUN.
```

2.9 Compile error using DISPLAY ADDRESS OF [data-item]

ID: 1145496936

A compatibility issue with the DISPLAY statement has been addressed.
The following can now be compiled:

```
DISPLAY " ADDRESS: " ADDRESS OF data-item.
```

Note: what is DISPLAY'ed is a pointer, which is not a USAGE DISPLAY data item, not the value stored in *data-item*.

2.9 Compile error with WRITE INVALID KEY on a SEQUENTIAL file

ID: 1145496944

Some COBOL compiler allow the user of the WRITE … INVALID KEY phrase on SEQUENTIAL files.

COBOL-IT now allows this phrase to be compiled.

2.9 Compiler error declaring a SWITCH with other SPECIAL-NAMES declarations

ID: 1145498360

In some situations, the usage of SWITCHes in SPECIAL-NAMES, along with other SPECIAL-NAMES declarations, would cause a compiler error.

This has been corrected.

2.8

2.8.10

New

2.8 Optimizations of the INITIALIZE verb

Optimizations have been made to the MOVE operations that are performed by the INITIALIZE verb in cases where target fields are USAGE DISPLAY NUMERIC, or USAGE NATIONAL.

To deactivate the optimization, use the –fno-optimize-move compiler flag.

2.8 VALUE clause in EXTERNAL fields

The VALUE clause is now accepted for fields described as EXTERNAL. The VALUE clause can only be applied in the first program allocating the field. When applied in programs other than the first program allocating the field, the VALUE clause is ignored.

2.8 -ffile-auto-external (default is on)

The –ffile-auto-external compiler flag affects the way that the compiler treats variables describing file-names for files described as EXTERNAL.

When a file is declared as EXTERNAL, if the file-name is indicated as a variable name, in an ASSIGN DYNAMIC [file-var] clause, then file-var should be declared as EXTERNAL. Note that variables declared as EXTERNAL must be declared as level 01 or level 77.

If [file-var] is not declared as EXTERNAL, then the default behavior of the COBOL-IT Compiler is to implicitly declare an external variable name, and assign it a name derived from the FD named in the SELECT clause.

The convention used is as follows:

Consider the statement:

SELECT myfile ASSIGN DYNAMIC file-var...

…

77 file-var pic x(8) value “customer”.

In this case, file-var is not declared as EXTERNAL, so COBOL-IT issues the following warning:

‘file-var’ declared imiplicitly EXTERNAL AS ‘FE_file_myfile_ASSIGN’ (-fno-file-auto-external to disable).

Creating the implicit name based on the file name guarantees that the programmer will be able to give different names to [file-var] in different programs, and that they will nonetheless share the same value.

As noted in the Warning message, this functionality may be disabled using the compiler flag '-fno-file-auto-external'. However, when disabling this functionality, be aware, that if you have separate programs sharing the same EXTERNAL file that also have file-var fields, then changes made between the programs will not automatically be shared.

If file-var is declared explicitly as EXTERNAL, then this condition does not apply.

2.8 ls-ignore-record-size:[Y/N]

The compiler configuration flag is-ignore-record-size determines whether or not trailing spaces are stripped from a line sequential file before writing it to disk.

Possible values: Y/N

Default :
is-ignore-record-size: N

When set to N-
The RECORD SIZE clause for LINE SEQUENTIAL files is used to determine the number of bytes written to disk on a WRITE statement. Trailing spaces are not stripped.

When set to Y-
The RECORD SIZE clause for LINE SEQUENTIAL files is ignored. Trailing white spaces are stripped before WRITEing a record to disk.

2.8 COB_SUNSTUDIO12

On sun SPARC solaris, when using C compiler sun studio 12. you must define the environment variable "COB_SUNSTUDIO12" to avoid the warning about obsoleted flag "xarch=generic64".

Example:

COB_STUDIO12 = “Y”; export COB_SUNSTUDIO12

2.8 OCCURS allowed with PIC 1 BIT

ID: 1145441794

Data items described as PIC 1 USAGE BIT are now allowed to have an OCCURS clause.

For example:

01 WS-COMM-BESTGRUPPE OCCURS 10000 PIC 1 BIT.
01 T-BIT OCCURS 16.
04 T-BEST PIC 1 USAGE BIT OCCURS 10000.

2.8 COMMIT and ROLLBACK implemented for VB/ISAM

COMMIT and ROLLBACK are now supported with the VB/ISAM file system.

2.8 COMMIT and ROLLBACK CALL use EXTFH driver function when -extfh is used

COMMIT and ROLLBACK are now supported with EXTFH drivers that support COMMIT and ROLLBACK, when the –extfh compiler flag is used.

2.8 MF Compatible CALL "x91" Function 15 is implemented

X”91” function 15 checks to see if a program exists.

Usage

```
call x"91" using fn-status,
                 x91-function,
                 param.
```

Parameters

fn-status	PIC X COMP-X
x91-function	PIC X COMP-X
param	Group Item containing
length-of-progname	PIC X COMP-X
program-name	PIC X(N).

Syntax

fn-status	set to program-name if successful, 0 if program not found.
X91-function	set to 15.
param	group item containing length of program-name, and program-name.

Code Sample

```
*
 ...
 77 FN-STATUS            PIC X COMP-X.
 77 X91-FUNCTION                PIC X COMP-X VALUE 15.
 01 PARAM.
   05 LWNGTH-OF-PROGNAME  PIC X COMP-X.
   05 PROGNAME            PIC X(6).
 ...
 PROCEDURE DIVISION.
     ...
     MOVE 6 TO LENGTH-OF-PROGNAME.
     MOVE “TESTIT” TO PROGRAM-NAME.
     CALL X"91" USING FN-STATUS, X91-FUNCTION, PARAM.
     ...
*
```

Fixes

2.8 -ffcdreg error on WRITE

ID: 1145474760

In some situations, FileName Length was not set after a WRITE operation.

This has been corrected.

2.8 Using a typedef in a USAGE clause with an OCCURS clause

ID: 1145464032

A typedef used in a USAGE clause could not contain an OCCURS clause if the OCCURS clause came after the USAGE clause.
This has been corrected.

2.8 INSPECT CONVERTING var-name TO ALL SPACES anomaly

ID: 1145459886

The DISPLAY of var-name, after CONVERTING TO ALL SPACES did not DISPLAY the spaces.

This has been corrected.

2.8 initialize-fd:yes doesn't work with FD EXTERNAL

ID: 1145443712

initialize-fd:yes was not applied when the FD was declared as EXTERNAL, as in:
FD myfile EXTERNAL.

This has been corrected.

2.8 LENGTH OF function not applicable in PERFORM statement

ID: 1145444702

The LENGTH OF function could not be applied during a PERFORM statement, as in:
PERFORM LENGTH OF Var1 TIMES
DISPLAY "ABC"
END-PERFORM.

This has been corrected.

2.8 VALUE clause with OCCURS

ID: 1145454240

Using the syntax VALUE SPACES with an OCCURS clause would only cause the first occurrence to be initialized to SPACES. For example, the statement:

01 WPRGINIT OCCURS 3 PIC X(256) VALUE SPACES.
Would only case PRGINIT(1) to be initialized to SPACES.

This has been corrected.

2.8 EXTERNAL field could crash debugger

When a field described as EXTERNAL declared in a COPY book was not used in the source, the debugger would crash if it tried to read the value of that field.

This has been corrected.

2.8 SELECT defined without EXTERNAL or DYNAMIC

ID: 1145451462

In a file which is not explicitly described as EXTERNAL or DYNAMIC, the phrase:
SELECT AF-F ASSIGN TO AF would return a compiler error if AF were a variable that had no value.

This has been corrected.

2.8 Compilation Error with option -g or -ftraceall

ID: 1145451404

On the X86 64-bit SUSE Linux platform, compiling with the –g or –ftraceall would create an error during the "C" compilation phase.

This has been corrected.

2.8 Remote compile crashs

ID: 1145449606
Develeoper Studio, version 1.4.2

In the Remote System perspective, the remote compile process could crash in some circumstances.

This has been corrected.

2.8 BIT order if you make a redefine on a PIC X value

ID: 1145447158

A REDEFINE on a PIC X that consists of 8 PIC1(1) USAGE BIT items would reproduce the BIT order incorrectly.

This has been corrected.

2.8 move-noninteger-to-alphanumeric:warning on MOVE SPACES TO NUMVAR

ID: 1145448640
When the Compiler Configuration Flag contains the setting:
move-noninteger-to-alphanumeric:warning

an "Error: Invalid MOVE statement" was incorrectly returned for the statement:

MOVE SPACES TO NUMERIC-VAR

This has been corrected.

2.8 RETURN on SORT causes memory fault when using external sort module

ID: 1145437242

The RETURN statement on a SORT could cause a memory fault, when using an external SORT module, and compiling with the –extfh and –extsm compiler flags.

This has been corrected.

2.8 READ NEXT LOSES CURRENT RECORD

ID: 1145437228

In series of READ NEXT/WRITE statements, it was possible for a DUPLICATE KEY file status on a WRITE statement, to cause the next READ NEXT statement to trigger an AT END condition.

This has been corrected.

2.8 EXTFH sets wrong lock mode on OPEN OUTPUT

ID: 1145427840

The COBOL-IT External File Handler (EXTFH) would incorrectly set the lock mode to 1 (Open Exclusive) on OPEN OUTPUT. Since the SELECT did not explicitly ask for an EXCLUSIVE lock, the lock mode should have been 0.

This has been corrected.

2.8 Referencing low-values on CLASS SPECIAL-NAMES caused compiler abort

ID: 1145436670, 1145442126

Referencing LOW-VALUES in the CLASS Clause in SPECIAL-NAMES would cause the compiler to abort. For example: SPECIAL-NAMES. CLASS WHITESPACE IS low-values.

This has been corrected.

2.8 ON EXCEPTION phrase caused memory fault for CALL statement

ID: 1145437552

In certain circumstances, having an ON EXCEPTION statement attached to a CALL statement, where the target of the CALL statement was a “C” routine, would cause a runtime memory fault.

This has been corrected.

2.8 USING EXTERNAL FILE HANDLER, BAD STATUS RETURN

ID: 1145438276

The COBOL-IT external file handler (EXTFH) reported a file status of 95(9/053) in when it should have returned a file status of 9E(0/069).
This has been corrected.

2.8 Compilation option SYSOUT

ID: 1145441944

When compiling a main program with –sysout=filename and a subprogram without the –sysout compiler flag, the subprogram would incorrectly DISPLAY into filename, and the main program would incorrectly DISPLAY onto the console after returning from the subprogram.

This has been corrected.

2.8 FILE DEFINED AS EXTERNAL CLOSED after CANCEL

ID: 1145441956

A file declared as EXTERNAL would be CLOSEd after the CANCEL of a subprogram.

This has been corrected.

2.8 INITIALIZE with ref mod causes invalid behavior or memory fault

ID: 1145442210

Using reference modification in an INITIALIZE statement would cause a SIGSEGV, attempt to reference unallocated memory, abnormal termination. For example,

INITIALIZE GROUP-ITEM (4:). (or)

INITIALIZE GROUP-ITEM (1:30).

This has been corrected.

2.8 Error compiling with -fstatic-call

ID: 1145444140

The compiler would abort when using the –fstatic-call compiler flag in a program with a CALL whose target was an ENTRY statement in the same program. If the –fstatic-call flag were not used, the compiler would not abort.

This has been corrected.

2.8 cobc -x -o .\object produced exe in current directory

ID: 1145445746

In version 2.7.3, the Windows compiler would incorrectly parse the command-line
>cobc –x –o .\object testit.cbl
The result would be that the executable would be created in the current directory, instead of in the .\object directory.

This has been corrected.

2.7

2.7.3

New

2.7 citmake

All platforms now include include now a copy of GNU Make renamed as citmake. Tthis is required by the Developer Studio IDE to build dependencies

2.7 first-tab-width: [integer]

The compiler configuration flag first-tab-width lets you define the size of the first tab.

Possible values: Any integer

Example:
First-tab-width: 8

Notes-

This is done to support the RM/COBOL standard that defines the first tab as 8 characters, and following tabs as 6 characters. To mimic this standard, you would set tab-width: 6, and first-tab-width: 8.

2.7 screen-exceptions: [Y/N]

The compiler configuration flag screen-exception mimics the behavior of the environment variable COB_SCREEN_EXCEPTIONS. .

Possible values: Y / N

Default:
screen-exceptions: N

When set to Y-

Causes the runtime to behave as if the environment variable COB_SCREEN_EXCEPTIONS=Y

Note- The compiler configuration flag screen-exceptions, when set to “Y”, enables use of the Page Up, Page Down, Up Arrow, and Down Arrow keys on Field-level ACCEPT statements.

When screen-exceptions:Y, the COBOL-IT runtime will return the CRT Status values as described in the table below. Note that Page Up, Page Down, Up Arrow, and Down Arrow are enabled by default when ACCEPTing a Screen:

Condition	Key Pressed	Cit Value returned to CRT-STATUS-VAR
On ACCEPT <Screen> or On ACCEPT <Field> where the	Page Up	2001

environment variable COB_SCREEN_EXCEPTIONS=Y or the compiler configuration flag screen-exceptions:Y		
	Page Down	2002
	Up Arrow	2003
	Down Arrow	2004
	Print	2006

2.7 screen-raw-keys: [Y/N]

The compiler configuration flag screen-raw-keys mimics the behavior of the environment variable COB_SCREEN_RAW_KEYS.. .

Possible values: Y / N

Default:
screen-raw-keys: N

When set to Y-

Causes the runtime to behave as if the environment variable COB_SCREEN_RAW_KEYS=Y

Note- The "raw keys" are the Home, End, Insert, Delete, and Erase EOL keys. When screen-raw-keys is set to Y, the COBOL-IT runtime will return CRT Status values as described in the table below:

Condition	Key Pressed	Cit Value returned to CRT-STATUS-VAR
Where the environment variable COB_SCREEN_RAW_KEYS=Y Or the compiler configuration flag screen-raw-keys:Y	Home	2007
	End	2008
	Ins	2009
	Del	2010
	Erase EOL	2011

2.7 -faccept-with-auto

The compiler flag –faccept-with-auto causes the WITH AUTO clause to be assumed by default on a field-level ACCEPT statement. When not compiling with –faccept-with-auto, the WITH TAB clause is assumed by default on a field-level ACCEPT statement. .

2.7 LIKE Clause

COBOL-IT now supports the LIKE Clause.

The LIKE clause allows you to define the PICTURE, USAGE, and SIGN characteristics of a data item by copying them from a previously defined data item

General Format

LIKE *data-name*

Where *data-name* can be an elementary, or group item.

Example:

```
01 FELD1 PIC X(10).
01 FELD2 LIKE FELD1.
*>In the example above,  FELD2 is described with the definition PIC X(10).


01 RECORD1.
   05 FELD1  PIC X(10).
   05 FELD2  PIC 9(4).

01 RECORD3 LIKE RECORD1.


*> In the example above, RECORD3 is described with the definition :
     01 RECORD3 .
        05 FELD1  PIC X(10).
        05 FELD2  PIC 9(4).
```

2.7 Newly supported DISPLAY syntax

The following DISPLAY syntax is now supported:

```
DISPLAY field1 LINE 1 COLUMN 3 Field2 AUTO UPDATE
```

Prior to this enhancement, each field had to be the target of a separate DISPLAY statement, and the AUTO UPDATE clause was supported as WITH AUTO UPDATE.

2.7 Newly supported ACCEPT syntax

The following ACCEPT syntax is now supported:

```
ACCEPT field1 ON EXCEPTION Field2  CONTINUE.
```

In this phrase, when an EXCEPTION key is pressed, the CRT STATUS is stored in Field2.

2.7 HIGH, LOW, NO BEEP keywords

HIGH is now recognized as equivalent to HIGHLIGHT in both ACCEPT and DISPLAY statements.

LOW is now recognized as equivalent to LOWLIGHT in both ACCEPT and DISPLAY statements.

NO BEEP is now supported, suppressing all warning beeps when input is performed.

2.7 CBL_ALLOC_DYN_MEM

CBL_ALLOC_DYN_MEM dynamically allocates memory, returning an addres (MEMORY-POINTER) and a size..

Usage:

```
call "CBL_ALLOC_DYB_MEM" using     memory-pointer
                       by value  memory-size
                       by value memory-flags
                       returning call-status
```

Syntax:

Memory-pointer is a data element described as USAGE POINTER. Memory-pointer must be described as an 01-level data item.

Memory-size represents the size of the memory being allocated, in bytes. Memory size is described as PIC x(4) comp-5.

Memory-Flags describe the type of memory being allocated. This is done by setting of bits on a 4-byte field. See the table below for guidelines on setting bits to describe a type of memory. Memory-flags is described as PIC x(4) comp-5.

Call-status is a return code.

2.7 CBL_ALLOC_MEM

CBL_ALLOC_MEM dynamically allocates memory, returning an addres (MEMORY-POINTER) and a size..

Usage:

```
call "CBL_ALLOC_MEM" using     memory-pointer
                       by value  memory-size
                       by value memory-flags
                       returning call-status
```

Syntax:

Memory-pointer is a data element described as USAGE POINTER. Memory-pointer must be described as an 01-level data item.

Memory-size represents the size of the memory being allocated, in bytes. Memory size is described as PIC x(4) comp-5.

Memory-Flags describe the type of memory being allocated. This is done by setting of bits on a 4-

byte field. See the table below for guidelines on setting bits to describe a type of memory. Memory-flags is described as PIC x(4) comp-5.

Call-status is a return code.

2.7 CBL_FREE_MEM

CBL_FREE_MEM frees memory allocated by the CBL_ALLOC_MEM routine.

Usage:

```
CALL  "CBL_FREE_MEM"  USING BY VALUE  memory-pointer
                      RETURNING call-status.
```

Syntax:

Memory-pointer is a data element described as USAGE POINTER, which has been populated with a value by the CBL_ALLOC_MEM routine.

Call-status is a return code.

2.7 CBL_FREE_DYN_MEM

CBL_FREE_DYN_MEM frees memory allocated by the CBL_ALLOC_DYN_MEM routine.
Usage:

```
CALL  "CBL_FREE_DYN_MEM"  USING BY VALUE  memory-pointer
                          RETURNING call-status.
```

Syntax:

Memory-pointer is a data element described as USAGE POINTER, which has been populated with a value by the CBL_ALLOC_DYN_MEM routine.

Call-status is a return code.

2.7 rmcobol.conf

A new standard configuration file, rmcobol.conf, is included with the standard distribution.

To access the new rmcobol.conf file, use the –std-‘rmcobol’ compiler flag.

Note- rmcobol.conf has been designed to reproduce RM/COBOL behaviors, with particular emphasis on the default CRT-STATUS mappings:

Fixes

2.7 Improved parsing of COPY statement:

The phrase : COPY file.ext. was incorrectly parsed. As a result, it was necessary to enclose the copyfile name in quotes, and use COPY "file.ext". COBOL-IT now supports the unquoted copyfile name.

2.7 TITLE/SKIP1/SKIP2/EjECT

In some cases, errors could occur when parsing TITLE/SKIP1/SKIP2/EJECT. This has been corrected.

2.7 VBISAM/Delete record bug

In some situations, a delete of a record in a VBISAM file could cause affect the ability to retrieve the record after the deleted record. This has been corrected.

2.7 Initialize field with SIGN SEPARATE

In some situations, an INITIALIZE of a field described with SIGN SEPARATE would cause the SIGN to disappear. This has been corrected.

2.7 CALL "SYSTEM" case-sensitive

CALL "SYSTEM" using …. Is now case-insensitive in operating environments that are case-insensitive.

2.6

2.6.14

New

2.6.14 key-dup-always-22

The compiler configuration flag key-dup-always-22 forces the runtime to return a file status of 22 on a duplicate key condition.

Possible values: yes, no

Example:
key-dup-always-22: yes

When set to yes,

When adding a record to an INDEXED file that is open in OUTPUT mode, if a duplicate key condition is detected, the runtime will return a file status 22.

2.6.14 Listing file includes memory mapping

The listing file produced by the –t compiler flag now includes memory mapping information. At the end of the listing, in lines that are commented, the list file now reports the size and offsets of the data fields in the Working-Storage Section.

2.6.14 -ffcdreg

The –ffcdreg compiler flag corresponds to the FCDREG compiler directive. This compiler flag provides functionality in applications that are using the EXTFH file system interface.

As background, EXTFH makes use of a standardized File Control Description (FCD), through which information passes to and from the EXTFH-compliant data source.
An FCD is created for each file that is mapped to an EXTFH-compliant data source.

It can be useful inside a program to directly read and write the FCD. The FCDREG compiler directive was developed for this purpose, and the COBOL-IT implementation of this functionality is the –ffcdreg compiler flag. When you compile with the –ffcdreg compiler flag, a register is created for each [filename] which is named “FH--FCD of [filename]". Note that there are two hyphens in the name “FH--FCD”. By describing the FCD structure, and positioning the beginning of the structure at the address of “FH--FCD of [filename]”, individual elements within the structure can be read and written.

Note- The FCD structure is described in a copy file called XFHFCD.CPY, which is included in the $COBOLITDIR\copy directory in Windows, and the $COBOLITDIR/share/config directory on UNIX/Linux-based systems.

For example:

1- Include a reference to the FCD in your Linkage Section, as follows:

```
LINKAGE SECTION.
 01 FCD.
 COPY "XFHFCD.CPY".
```

2- Sync the address of FCD with the address of FH--FCD OF FIL1.

```
PROCEDURE DIVISION.
. . .
   SET ADDRESS OF FCD TO ADDRESS OF FH--FCD OF FIL1.
```

3- After performing the SET statement above, the fields in XFHFCD.CPY can be read and written.

2.6.14 -freplace-additive

The –freplace-additive compiler flag allows for the use of the REPLACE ADD verb, which has the effect of nesting a REPLACE statement inside an existing REPLACE statement. Nested REPLACE statements are executed before outer REPLACE statements in COBOL-IT's precompile phase. Note that a a REPLACE stack can be cleared with the REPLACE OFF statement.

2.6.14 REPLACE ADD

The REPLACE ADD verb nests a REPLACE statement inside an existing REPLACE statement.

2.6.14 CALL Syntax enhaanced

The CALL syntax now allows the GIVING/RETURNING clause to precede the USING clause. Berkeley DB 5.2 supported in XA Mode COBOL-IT has been certified with Berkeley DB 5.2 running in XA Mode.

2.6.12 COB_SCREEN_UPDATE_FIRST_KEY_ERASE

- The COB_SCREEN_UPDATE_FIRST_KEY_ERASE environment variable, when set to Y, causesall field-level ACCEPT WITH UPDATE statements to behave as described below:

  If the first key pressed when entering the field is an alpha/number key the field is erased, and the keystroke recorded. Reformatting rules are applied when exiting the field if the number of characters entered is fewer than can be held by the field.

  Consider an example:

  A field-level ACCEPT WITH UPDATE is being done on a numeric data element described as PIC 9(4). The user enters a single digit "1", and exits the field. The environment variable COB_SCREEN_UPDATE_FIRST_KEY_ERASE is set to "Y".

  Prior to recording the keystroke, the runtime initializes the field to zeroes. Then, the runtime records the keystroke of "1" in the first character position, then exits the field.

  In many (perhaps most) cases, the user will expect the data item to reformat from 1000 to 0001. This reformatting is the default behavior when COB_SCREEN_UP-DATE_FIRST_KEY_ERASE is set to "Y".

For the cases where this reformatting is not desired, the user can disable this default behavior with the COB_SCREEN_DISABLE_REFORMAT behavior, as described below.

2.6.12 COB_SCREEN_DISABLE_REFORMAT=Y

- The COB_SCREEN_DISABLE_REFORMAT environment variable, when set to Y, disables the reformatting associated by default with the COB_SCREEN_UPDATE_FIRST_KEY_ERASE behavior.

2.6.12 -faccept-with-update

- The –faccept-with-update compiler flag causes field-level ACCEPT statements to be interpreted as containing the "WITH UPDATE" clause.

2.6.12 accept-with-update:yes | no

- The accept-with-update compiler configuration flag, when set to "yes", causes all field-level ACCEPT statements to be interpreted as containing the "WITH UPDATE" clause.

2.6.12 COB_SCREEN_EXCEPTIONS

- The environment variable COB_SCREEN_EXCEPTIONS, when set to "Y", enables use of the Page Up, Page Down, Up Arrow, and Down Arrow keys on Field-level ACCEPT statements.
- When COB_SCREEN_EXCEPTIONS=Y, the COBOL-IT runtime will return the CRT Status values as described in the table below. Note that Page Up, Page Down, Up Arrow, and Down Arrow are enabled by default when ACCEPTing a Screen:

Condition	Key Pressed	Cit Value returned to CRT-STATUS-VAR
On ACCEPT <Screen> or On ACCEPT <Field> where COB_SCREEN_EXCEPTIONS=Y	Page Up	2001
	Page Down	2002
	Up Arrow	2003
	Down Arrow	2004
	Print	2006

2.6.12 COB_SCREEN_ESC

- The environment variable COB_SCREEN_ESC, when set to "Y", enables use of the escape key. When COB_SCREEN_ESC=Y, the COBOL-IT runtime will return the CRT Status value as described in the table below:

Condition	Key Pressed	Cit Value returned to CRT-STATUS-VAR
Where COB_SCREEN_ESC=Y	Esc	2005

2.6.12 COB_SCREEN_RAW_KEYS

- The environment variable COB_SCREEN_RAW_KEYS, when set to "Y", enables use of the "raw keys".
- The "raw keys" are the Home, End, Insert, Delete, and Erase EOL keys. To enable use of the "raw keys", set the environment variable COB_SCREEN_RAW_KEYS=Y. When COB_SCREEN_RAW_KEYS=Y, the COBOL-IT runtime will return CRT Status values as described in the table below:

Condition	Key Pressed	Cit Value returned to CRT-STATUS-VAR
Where COB_SCREEN_RAW_KEYS=Y	Home	2007
	End	2008
	Ins	2009
	Del	2010
	Erase EOL	2011

2.6.12 crtstatus-map: cit-value user-value

- It is possible to create your own CRT STATUS Map, using the compiler configuration flag crt-status-map as described below.

- When crt-status-var is declared as UNSIGNED-INT, the runtime copies user-value directly into the crt-status-var. For this case, if you wish to remap the first four function keys, from CIT-Values of 1001 through 1004 to single digits 1 through 4, you would add the entries:

```
crtstatus-map: 1001 1
crtstatus-map: 1002 2
crtstatus-map: 1003 3
crtstatus-map: 1004 4
```

- The use of hex notation (x00 through x127) to describe characters in user-value is supported. For this case, if you wish to remap the first four function keys, from CIT-Values of 1001 through 1004 to single digits 1 through 4, you would add the entries:

```
crtstatus-map: 1001 x31
crtstatus-map: 1002 x32
crtstatus-map: 1003 x33
crtstatus-map: 1004 x34
```

- Hex notations permit the user to use values that fall outside of the normal alphanumeric range. More than one character can be represented using hex notation. For example, if you wish to map the cit-value of 1004 to the user-value of 1234 using hex notation, you would add the entry:

crtstatus-map 1004 x31323334

- In the example above, in the description of the user-value, after the “x”, the values “31”, “32”, “33”, “34” are concatenated to represent the string “1234”. When using hex notations, it is recommended that the receiving field be described as PIC X(4).
- If no crtstatus-map is defined , CRT STATUS values are converted to PIC 9(4) and copied into the crt-status-var.

2.6.12 -fstatus-map cit-value user-value

- The usage of the –fstatus-map compiler flag parallels the usage of the crtstatus-map compiler configuration flag, Multiple iterations may be included on the same compiler command line.
- To compile the program hello.cbl using the –fstatus-map compiler flag to re-map the default CRT-STATUS of 1001 to 1, use the command:

  cobc –fstatus-map 1001 1 hello.cbl

2.6.12 NUMERIC [SIGN IS | SIGN | IS] [LEADING | TRAILING] [| SEPARATE]

- The COBOL-IT Compiler Suite now supports the NUMERIC SIGN clause in SPECIAL-NAMES.
- The NUMERIC SIGN clause is used to define a default sign storage convention, when the sign storage for a data item is not specified. The COB_SCREEN_INPUT_INSERT_TOGGLE environment variable, when set to 1, causes the INS Key to toggle between Overwrite and Insert mode. The default behavior of the INS key is to insert a SPACE at the current cursor position.

2.6.12 SELECT ... ASSIGN DYNAMIC myfield

- The COBOL-IT Compiler Suite now supports the ASSIGN DYNAMIC clause in the SELECT phrase.
- The ASSIGN DYNAMIC phrase permits the association of a file-name with a working-storage variable, which may be assigned dynamically at run time. If the file name (myfield in the example above), is not declared in the Working-Storage Section, it will be automatically declared.
- This behavior was formerly supported when using the compiler configuration file setting assign-clause:MF. With this enhancement, the ASSIGN DYNAMIC behavior can be applied regardless of the setting of the assign-clause compiler configuration flag.

2.6.12 Berkeley DB EXTFH file handler

- The Berkeley DB EXTFH file handler is now provided in the Enterprise edition, for use with Berkeley DB version 4.7. End User License Agreements for the Berkeley DB must be obtained separately by the user from their provider of Berkeley DB.

2.6.12 Citsort may now use user defined EXTFH drivers

- Using the COBOL-IT Dynamic EXTFH mapping, the user must define:
  - COB_EXTFH_INDEXED=<Indexed EXTFH Function>
    or
    COB_EXTFH_FLAT=<Flat file EXTFH Function>
    or
    COB_EXTFH=<all purpose EXTFH Function>
  - Then user must define the library to load the EXTFH Function
    COB_EXTFH_LIB=<path to the dll/shared library holding EXTFH function>

Example for use Berkeley DB with CITSORT:
COB_EXTFH_INDEXED=BDBEXTFH
COB_EXTFH_LIB=/opt/cobol-it/lib/libbdbextfh.so

2.6.7 Remote debugging

o Cobcdb is now enhanced to provide the new IDE 1.3 remote debugging capabilities.

2.6.7 Global DEBUG-ID

- Reverse attaching for debugging may now use a DEBUG-ID instead of PID (Process ID).
- Before running the program to debug, define the environment variables :export COB_DEBUG_ID=<integer value>
- Then from any other console, you may catch the running program using his DEBUG-ID instead of the PID:deet –p <debug-id>

2.6.7 CALL "C$DEBUG" USING DEBUG-ID

- When using the debugger inline break point "C$DEBUG", you may now add an numeric parameter that will be used as DEBUG-ID.

- Then from any other console, you may catch the running program using his DEBUG-ID instead of the PID:deet –p <debug-id>

2.6.3 line-seq-recording-mode

- The line-seq-recording-mode configuration file flag affects the treatment of the RECORDING MODE F with LINE SEQUENTIAL files.
- # Value: Boolean
- Default value: NO
- If set to NO, the clause is ignored, and trailing spaces are removed from the record before writing it to disk.
- If set to YES, the record is written to disk with trailing spaces.

2.6.3 -finitialize-fd

- The –finitialize-fd compiler flag causes the program to initialize all records declared in the File Section upon startup.

2.6.3 CALL-CONVENTION for ENTRY declarations

- CALL-CONVENTION is not supported with ENTRY declarations/

2.6.0 Faircom c-Tree ACE.

- COBOL-IT now supports Faircom's C-Tree ACE database engine. Please refer to the "Reference Manual" for more details.

2.6.0 -fprinter-crlf

- The –fprinter-crlf compiler flag can be used to assign CR/LF as the End-of-Record delimiter that is sent to the printer on files described with an ASSIGN TO PRINTER clause in their SELECT statement, regardless of the underlying operating system.

2.6.0 SW0-SW15

- SW0 to SW15 are now supported as aliases of SWITCH-1 to SWITCH-16

2.6.0 ACCEPT Screen or Field

New environment variables have been created to control the runtime behavior of ACCEPT Screen or Field.

2.6.0 COB_SCREEN_INPUT_UNDERLINED

- The COB_SCREEN_INPUT_UNDERLINED environment variable, when set to 1, causes all input fields to be underlined.

2.6.0 COB_SCREEN_INPUT_REVERSED

- The COB_SCREEN_INPUT_REVERSED environment variable, when set to 1, causes all input fields to be reversed.

2.6.0 COB_SCREEN_INPUT_BOLDED

- The COB_SCREEN_INPUT_BOLDED environment variable, when set to 1, causes all input fields to be displayed in bold.

2.6.0 COB_SCREEN_INPUT_FILLER [char]

- The COB_SCREEN_INPUT_FILLER environment variable, when set to [char], changes the PROMPT character to [char].

Fixes

2.6.14 -E output anomaly

The –E compiler flag preprocesses only, and returns output to stdout. When using -E compiler flag, EXEC... END-EXEC statements were not included in the preprocessed output.

This has been corrected.

2.6.12 Unmatched END-EXEC statements

- Unmatched END-EXEC statements now cause an error. Formerly, this was a Warning condition.

2.6.12 The debugging pattern *#9999 "filename"...

- The debugging pattern *#9999 “filename” is now changed amd more complex. it is now : *#DEBUG*9999* GUBED "Filename"

2.6.12 ACCEPT Field AT 1010

- ACCEPT Field AT 1010 is now correctly parsed as ACCEPT field AT LINE 10 COLUMN 10.

2.6.12 DISPLAY SPACE AT 1010

- DISPLAY SPACE AT 1010 now erases the rest of the screen starting at LINE 10 COLUMN 10

2.6.12 Berkeley DB EXTFH File Handler

- Several fixes have been applied to the Berkeley DB EXTFH file handler, and support for XA compliance.

2.6.12 Line Sequential files with binary values < x20.

- It was possible to get errors in the READing of line sequential files containing binary values < x20. This has been fixed.

2.6.7 CitSORT

- Several fixes have been applied to CitSORT when OMMIT/INCLUDE Conditions and SUM FIELD.

2.6.7 EXTFH interface

- The EXTFH interface has been improved to be more conformant to MF behaviors when using OPTIONAL files.

2.6.7 BDB EXTFH interface

- The Berkeley DB EXTFH interface has been improved to support XA compliant

2.6.7 Remote debugging

o Cobcdb is now enhanced to provide the new IDE 1.3 remote debugging capabilities.

2.6.7 Global DEBUG-ID

- Reverse attaching for debugging may now use a DEBUG-ID instead of PID (Process ID).
- Before running the program to debug, define the environment variables :export COB_DEBUG_ID=<integer value>
- Then from any other console, you may catch the running program using his DEBUG-ID instead of the PID:deet –p <debug-id>

2.6.7 CALL "C$DEBUG" USING DEBUG-ID

- When using the debugger inline break point "C$DEBUG", you may now add an numeric parameter that will be used as DEBUG-ID.
  Then from any other console, you may catch the running program using his DEBUG-ID instead of the PID:deet –p <debug-id>

2.6.3 -fthread-safe on Windows

- The –fthread-safe compiler flag generates code that requires Windows Vista, or later Windows releases in order to run in true thread-safe mode. This would include Windows 7, and Server 2008. Now, if code is compiled with the –thread-safe compiler flag on versions of Windows released prior to Windows Vista, a warning is generated, which states that the code requires Windows Vista, or later in order to run truly thread-safe.

2.6.3 libextsm

- Several fixes have been applied to the EXTSM interface.

2.6.3 CitSORT

- Several fixes have been applied to CitSORT when sorting LINE SEQUENTIAL files.

2.6.3 COB_SCREEN_INPUT_INSERT_TOGGLE

- The COB_SCREEN_INPUT_INSERT_TOGGLE environment variable, when set to 1, causes

the INS Key to toggle between Overwrite and Insert mode. The default behavior of the INS key is to insert a SPACE at the current cursor position.

2.6.3 DEL Key Behavior

- Pressing the DEL key deletes the current character and moves the cursor one character to the left. A SPACE is inserted at the end of the field.

2.6.3 EOL Key Behavior

- The EOL key erases to the end of the line.

2.6.3 BACKSPACE Key Behavior

- The BACKSPACE key shifts the contents of the field one character to the left, beginning at the current character.

2.6.0 PROCEDURE USING in WORKING -STORAGE

- When a field or a group declared in WORKING-STORAGE is used in the USING clause of the PROCEDURE DIVISION, the WORKING-STORAGE data item is populated when the program initiates. Subsequent efforts to update the data item would fail. This has been corrected.

2.6.0 START NOT >... READ PREVIOUS

- In the Sequence START NOT > followed by READ PREVIOUS, the first returned record was not always correct. This has been corrected.

2.6.0 CANCEL

- A circumstance was identified in which the CANCEL verb would appear to execute successfully, but would not unload the target program from memory, when the FULL_CANCEL environment variable was being used.

  As an example:

Program SAMPLE_1 contains:	CALL "SAMPLE_2"
	CANCEL "SAMPLE_2"
Program SAMPLE_2 contains:	CALL "SAMPLE_3"
	CANCEL "SAMPLE_3"

  Program SAMPLE_3 is a stub program, which returns directly.

  In this case, SAMPLES_2 was not unloaded from the memory.
  This problem is now fixed.

2.5

Object code produced with version 2.5.x may not be used with the 2.4.x runtime.The target runtime must also be updated to version 2.5.x.

2.5.11

New

2.5.11 Tab expansion in LINE SEQUENTIAL files

- When reading a LINE SEQUENTIAL file :
  If the configuration parameter "line-seq-mf" is true (the default), the tab characters not preceded by a 0 are expanded to spaces. The tab stop is fixed to 8 characters.

2.5.10a Getting Started

- The "Getting Started with the COBOL-IT Compiler Suite" document has been revised and partially rewritten.

2.5.10a TYPEDEF

- The TYPEDEF field attribute is now supported.
- The TYPEDEF field attribute allows you to define a "user-defined" USAGE.
- Implementation has been made compatible with the MF implementation of the TYPEDEF clause.
- See documentation about the –f-global-typedef compiler flag for more details.

General Format
Level Name …{general definition} IS TYPEDEF …

Syntax
Level must be 1 or 77.
When the data element is a group-item, only the level 01 needs to be described with the **IS TYPEDEF** clause.
A data element described as **IS TYPEDEF** may not be used as storage.

2.5.10a $IF xxx DEFINED

- The conditional compilation test $IF xxx DEFINED is now supported. The test is TRUE if xxx is defined as a constant.

2.5.8 CALL-CONVENTION

- The CALL-CONVENTION syntax can be used to control which mechanisms the COBOL program should use when calling a subprogram, or which mechanisms it expects to have been used by the program which calls it. Each calling convention you want to use should be declared in the Special-Names paragraph. Here you can assign one of the call convention numbers, defined below, to a user-defined mnemonic name for later use.

General Format

[CALL-CONVENTIONnumber IS mnemonic]

Syntax

- **number** is a numeric literal representing the call convention.
  The call convention number is a 16-bit number defined as follows:

Bit	Meaning
0-1	Unsupported
2	= 0 - RETURN-CODE is updated on exit = 1 - RETURN-CODE is not updated on exit
3	= 0 –CALL is resolved dynamically at run time = 1 –CALL is resolved statically at compilation time
4-5	Unsupported
6	Windows Only = 0 –CALL use standard C call conventions = 1 –CALL use “STDCALL” WINAPI call conventions (used to call Windows standard API)

- Typical values are :
- 4 Do not modify RETURN-CODE. This is very useful when using Oracle
- Pro*Cobol. CALLs to the Oracle runtime do not modify RETURN-CODE.
- 72 Windows API Call.
- **Mnemonic** is a user-defined word used just after the CALL verb to modify the current call convention.

Code sample

```
IDENTIFICATION DIVISION.
PROGRAM-ID.     prog2.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
   CALL-CONVENTION 4 IS VOIDCALL.

   ...

PROCEDURE DIVISION.
A01.
   MOVE 19 TO RETURN-CODE.
   CALL VOIDCALL "callee".
*RETURN-CODE remains equal to 19 no matter what callee returns
```

2.5.8 defaultcall

- The defaultcall configuration file flag sets the default CALL-CONVENTION when no CALL-CONVENTION is named in a CALL statement.
- # Value: integer
- Default value: 0
- See SPECIAL-NAMES, CALL-CONVENTION for details on the usage of CALL-CONVENTION.

2.5.8 utf16-le

- The utf16-le configuration file flag, when set to “yes”, causes fields declared as PIC N to be stored in UTF16-LE (Little Endian) format. The (default) alternative is that fields declared as PIC N are stored in UTF16-BE (Big Endian) format.
- # Value: 'yes', 'no'
- Default : “no”

2.5.8 ls-utf16

- The ls-utf16 configuration file flag, when set to “yes”, causes line sequential files to be read, and stored internally in UTF16 format. UTF16-LE or UTF16-BE format is used, depending on the setting of the configuration file flag utf16-le.
- # Value: 'yes', 'no'
- Default value: no
- Note that the default data storage for data stored in UTF16 format is UTF16-BE.
- ls-utf16 may also be set in source using the meta comment $SET before the SELECT statement
- • $SET LSUTF16 Activate UTF16 storage
- • $SET NOLSUTF16 Deactivate UTF16 storage
- When UTF16 Storage is active, end-of-record is read/written in UTF16 format according to the setting of the utf16-le compilation flag (X’000A’ for BE and X’0A00’ for LE)

2.5.6 Runtime support for EXTFH

- Runtime environment variables COB_EXTFH, COB_EXTFH_INDEXED, COB_EXTFH_FLAT, and COB_EXTFH_LIB allow for the detection of an EXTFH interface at runtime.
- With this enhancement, you can make use of an EXTFH data source without having to compile with the –use-extfh compiler flag.

At run time define :

For EXTFH interface to RDBMS:
COB_EXTFH=<your extfh name>

For EXTFH interface to Indexed Files:
COB_EXTFH_INDEXED=<your extfh name>

For EXTFH interface to Sequential or Relative Files:
COB_EXTFH_FLAT=<your extfh name>

COB_EXTFH_LIB=<list of shared libscontainingextfh code>

For example :

COB_EXTFH=CTEXTFH

COB_EXTFH_LIB=/opt/mytools/lib/liba.so:/opt/mytools/lib/libb.so

2.5.6 ACCEPT/DISPLAY may be redirected at compilation time

- -sysin=<file used for input>
- –sysout=<file used for output>

For example: cobc –sysin=sysin.txt –sysout=sysout.txt testit.cbl

- <file used for input> must be a line sequential file.
- < file used for output> is created as a line sequential file.

2.5.6 Conditional compilation switch X0 – X9

- Switch X0 to X9 may now be used with conditional compilation $SET $IF
- Values are ON or OFF, default is OFF

Sample :

```
$SET X0=ON
…
$IF X0=ON
    DISPLAY "…."
$END
…
```

2.5.6 HP UX on PARiSC

- The COBOL-IT Compiler Suite Enterprise Edition distribution is now available HP UX 11.11 on PARiSC.

2.5.6 SIGN LEADING and/or SEPARATE

- New flags let you define SIGN IS LEADING or SIGN IS SEPARATE as default.

```
–fsign-leading  to make LEADING the default
–fsign-separate to make SEPARATE the default
```

2.5.4 Support for embedded c-TreeACE database engine

- The COBOL-IT Compiler Suite Enterprise Edition distributions will soon embed FairCom's c-treeACE database engine, providing access to a robust indexed file system, and a powerful set of utilities. Preliminary work has been done for that purpose.
- The c-treeACE database engine will be included in the coming release 2.6.0.
- Note that usage of the c-treeACE database engine will require a separate license. If you would like more information, please contact sales@cobol-it.com.

2.5.4 Solaris 10 on x86 system

- The COBOL-IT Compiler Suite Enterprise Edition distribution is now available for Solaris 10 on x86_64 systems.

2.5.3 APPLY clause in I-O CONTROL recognized

- The APPLY clause is recognized in the I-O CONTROL SECTION. However it is ignored.
    - Example: APPLY WRITE-ONLY on [file-name]

2.5.3 Using cobcrun and deet with Oracle (Windows)

- In Windows environments, the CALL statements generated by the precompiling process are resolved in calls to DLLs, which are provided by Oracle and installed on the Oracle client workstation. In Windows environments, it is not necessary to rebuild cobcrun, and cobcdb, as the CALL statements are resolved dynamically.
- Using deet with applications that make CALLs to Oracle libraries:
    - In Windows environments, generate a single dynamically loadable module (DLL) that includes the SQL library (orasql11.lib for Oracle 11) provided by Oracle, as in the example below.

- The following script creates procobdemo.dll, which can then be executed with the command “deetprocobdemo”. This example presumes that procobdemo.pco has already been precompiled, producing procobdemo.cbl as the output file.

```
set ICHOME=C:\INSTANTCLIENT_11_2
set ICLIBHOME=%ICHOME%\sdk\lib\msvc
set SQLLIB_lib=orasql11.lib

cobc –g –conf=myconf.conf –c procobdemo.cbl –o procobdemo.obj
cobc –b procobdemo.obj %ICLIBHOME%\%SQLLIB_lib%
```

2.5.3 BUILDING cobcrun and deet for Oracle (Linux/Unix)

- On Linux/Unix machines, runtime access to Oracle subroutines requires that the Oracle libraries be re-linked with cobcrun, and debugger access to Oracle subroutines requires that Oracle libraries be re-linked with cobcdb.
- Rebuilding cobcrun to include Oracle libraries:
  - To build your own cobcrun including an Oracle CALL entry point (Linux):

```
cobc -x -flink-only -o cobcrun
$COBOLITDIR/lib/cobol-it/cobcrun.o
$ORACLE_HOME/precomp/lib/cobsqlintf.o -L $ORACLE_HOME/lib/
-lclntsh
```

Replace the existing cobcrun with the newly created cobcrun, and make sure it is in your PATH.

- Rebuilding cobcdb to include Oracle libraries:
- Note-To debug a COBOL executable that has been linked with Oracle libraries, you needto link the same Oracle libraries into the debugger launcher (cobcdb).
  - To build your own cobcdb including an Oracle CALL entry point (Linux/Unix):

```
cobc -x -flink-only -o cobcdb
$COBOLITDIR/lib/cobol-it/cobcdb.o –lcitsupport
$ORACLE_HOME/precomp/lib/cobsqlintf.o -L $ORACLE_HOME/lib/
-lclntsh
```

Replace the existing cobcdb with the newly created cobcdb, and make sure it is in your PATH.

2.5.2 Support for PIC 1(n) added

- The PICTURE PIC 1(n) is now supported. The following General Rules apply:
  - Each “1” represents one BIT
  - PIC 1 may be used with USAGE DISPLAY, COMP, COMP-5, COMP-X, BIT
  - The maximum number of bits allowed is 64
  - Display of a PIC 1 field will display the bits, for example, “000101”.
  - When used with USAGE COMP, COMP-5, COMP-X the minimum number of bytes

needed to store the required number of bits is allocated.

- When used with USAGE DISPLAY, one byte per bit is allocated. Each byte contains "0" or "1".
- PIC 1 fields are numeric and may be used in computations and MOVEs for their integer value.
- When using USAGE BIT, the exact numbers of bits are used that are allocated in the current bit field. For example, the following declaration allocates 3 bytes:

```
01 bx PIC 1(24) USAGE COMP-5.
```

- Literary binary constants are now supported, for example VALUE B'1011'.

2.5.2 Support for USAGE BIT added

- The USAGE BIT Clause is now supported. The following General Rules apply:
  - USAGE BIT allocates a field as a bit field.
  - PIC 1(n) is the only picture allowed.
  - USAGE BIT may not redefine or be redefined.
  - OCCURS may not be used WITH USAGE BIT
  - Consecutive USAGE BIT fields are packed into the same byte area with a maximum size of 8 bytes (64 Bits).
  - If a USAGE BIT field does not fit in the current byte area because the resulting byte area would be greater than 64 bits, then a new byte area will be allocated. Remaining unused bits of a byte area are left unused.
  - As an example, a single byte would be allocated to contain these 3 fields b1, b2, and b3.

```
03 b1 PIC 1 USAGE BIT.
03 b2 PIC 1(3) USAGE BIT.
03 b3 PIC 1(4) USAGE BIT.
```

  - As an example, a single byte would be allocated to contain fields b4 and b5 in the example below, and 8 bytes would be allocated for b6:

```
03 b4 PIC 1 USAGE BIT.
03 b5 PIC 1(5) USAGE BIT.
03 b6 PIC 1(60) USAGE BIT.
```

  - The byte area stores bits in memory in "System Native" bit order (like COMP-5).

2.5.1 File compression is now supported in VBISAM.

- To enable compression in VBISAM, you must add :

```
$SET DATACOMPRESS "x"
```

before the SELECT statement.

"x" is a numeric integer literal, ranging from 1 to 9. Higher values produce better compression, at the expense of performance. A setting of 1 provides the best performance, a setting of 9 the best compression.

All users of an indexed file created with compression must re-compile their programs with the DATACOMPRESS compiler directive and use the same compression

setting.

To disable compression (after having enabled it)

```
$SET NODATACOMPRESS
```

2.5.0 Split keys are now supported in VBISAM

- *In the SELECT Phrase*
- *[RECORD KEY IS record-key [=key-segment . . .]]*
- *[ALTERNATE RECORD KEY IS alt-rec-key [=key-segment . . .]*
- *[WITH [NO] DUPLICATES]]*

- Example from a SELECT phrase
  - record key is split-key = field-4, field-1 and/or
  - alternate key is split-key = field-3, field-2 with duplicates
- Split keys provide the ability to describe record keys composed on non-contiguous segments in the file.
- When describing a split key, key-segement must be a field described in the FD.
- When describing a split key, *record-key*or*alt-rec-key*is a unique user-defined word not described in the Data Division. This user-defined word is referenced as [keyname] in the
  - READ KEY IS [key-name],
  - START [FILE] KEY [operator] [key-name]
  - MOVE [figurative constant] to [key-name] phrases.

2.5.0 CitSORT support VB/ISAM files

- The tool may be used to import/export VB/ISAM file from/to SEQUENTIAL file or to add, change and remove indexes of INDEXED VSAM FILES.

Example: To export a VB/ISAM file into a line sequential file:

>citsort use pres record f 83 org ix key (1,2,p) give pres2.txt org ls

Updates to the syntax include:

Organization Clause: org ix is now supported.

Key Clause:

Usage: [KEY ({start-pos,length,type}...)]

Syntax

Start-pos	The starting position of the record key, where the first byte has position 1.
Length	The length of the key, in bytes
Type	The key type, as selected from the table below:
	P Primary key

A	Alternate key
AD	Alternate key with duplicates
C	Component of the last-specified primary or alternate key.

2.5.0 fstatus-map

- The fstatus-map configuration file flag allows the mapping of COBOL-IT file IO status values to custom values.

Usage:

```
fstatus-map:  cit-value [=] custom-value
```

Example :fstatus-map: 22=67

In the example above, when the file system returns status 22, the value will be translated to 67 before being returned to the application.The fstatus-map flag may be repeated to describe multiple IO status mappings.

2.5.0 Profiling produces CPU and real elapsed times

The output produced by the profiler (when compiling with –fprofiler) now includes separate counts for CPU and real elapsed times.

2.5.0 move-to-group-separated

- The configuration file flag move-to-group-separated affects the spaces and zeroes are handled when moved to a group item when compiling the instructions:
  - MOVE SPACE TO GRP-XXX.
  - MOVE ZERO TO GRP-XXX.

    where GRP-XXX is a group item.

  - If set to **yes** , the MOVE treats the group item as the multiple fields and sub-fields described by the group.
  - If set to **no** (default) the MOVE treats the group item as a single alphanumeric (PIC X) field whose size is calculated as the aggregate size of all of the fields and sub-fields described by the group.

    This is useful when used with the compilation flag spzero.

Fixes

2.5.11 CitSORT

Several fixes have been applied to CitSORT.

2.5.10a PIC N

- A bug was detected in the MOVE of PIC N data items described with reference modification. The phrase MOVE N(1:3) TO X, when N is describe as a PIC N data type, and X is described as a PIC X data type now correctly MOVEs 6 bytes of information.

2.5.10a EXTFH interface

- The EXTFH interface has been improved to be more conformant to MF behaviors.

2.5.10a CitSORT

- Errors could occur in CitSORT when parsing split keys. This has been fixed.

2.5.8 cobmf on Windows

- cobmf (The MF command line emulator) is now working on Windows.

2.5.8 cobc fixes

- Several minor fixes, including the increasing of the command line's buffer size, have been apply to cobc.

2.5.8 CURSOR IS ...

- When using SPECIAL NAMES CURSOR and SCREEN SECTION, for example:

```
SPECIAL-NAMES.

   CURSOR IS CURSOR-POSITION,
   …
```

If the program changes the value of CURSOR-POSITION, the next ACCEPT field or screen will move cursor to the required position.

When preforming an ACCEPT SCREEN, the cursor (input) is moved to the best matching field.

2.5.6 EXTFH interface

- The EXTFH interface has been improved to be more conformant to MF behaviors.

2.5.6 ACCEPT/DISPLAY with DEET

- The DEET debugger could behave incorrectly when processing some ACCEPT/DISPLAY statements. This has been fixed.

2.5.4 EXTFH interface

- The EXTFH interface has been improved to be more conformant to MF behaviors.

2.5.4 LENGTH OF field

- In certain situations, where REDEFINES and OCCURS DEPENDING ON clauses were used together, LENGTH OF could be incorrectly computed. This has been fixed.

2.5.4 MOVE -0 Fixed to 0

- When a COMP-3 or USAGE DISPLAY source field containing a -0 is MOVEd, the target field will now store the value in the target field as +0.

2.5.4 COPY / REPLACING

- COPY xxx REPLACING handling has been improved.

2.5.3 RECORDING MODE F with LINE SEQUENTIAL files

- LINE SEQUENTIAL files declared with RECORDING MODE F are now supported. LINE SEQUENTIAL files declared with RECORDING MODE F are stored with fixed record length. Trailing spaces are not removed.

2.5.3 -L and –l Usable on Windows

- –L <Library path> and –l <library_name>are now useable on Windows. <library_name> must be the complete library name including the .LIB extension.

2.5.3 -b on Windows producesa .LIBfile in along with .DLL

- When producing a .DLL with the compiler flag –b in Windows operating environments, the .lib file, including stub entry point is now produced.

2.5.3 Dynamic SYSTEM and C$ call

- Dynamic callsof SYSTEM and C$ internal functionsarenow supported.
    - Example :77 function-call pic x(6) value “SYSTEM”.
    - …
    - CALL function-call.

2.5.3 Minor fix in INDEXED Files

- In VBISAM, some Delete/Read Next sequences, caused the wrong record to be returned. This has been fixed.

2.5.2 Bug using PIC 9(n)P(x)

- A memory allocation bug was introduced in 2.5.1 when using data items described as PIC 9(n)P(x). This has been fixed.

2.5.1 ACCEPT/DISPLAY syntax parsing

- A defect was detected in the parsing of ACCEPT/DISPLAY syntax. This has been fixed.

2.5.1 EXTFH handling

- There has been a minor fix in the EXTFH handling function.

2.5.1 Deet behavior on NEXT statement entering an Exception Routine

- If executing a “Next“ statement entered in an exception routine, the debugger would break at the entry-point to the exception routine. This has been fixed.

IMPORTANT Fix

2.5.1 Addition error with integer constant and signed USAGE DISPLAY data element

- The addition of an integer constant to a signed USAGE DISPLAY data element with decimal notation could give a FALSE result.
- As an example, the following code adds an integer constant (1) to the signed USAGE DISPLAY data element zman. In this scenario, the runtime would incorrectly calculate the result of the addition as 1.2, instead of .8. This has been fixed.

```
WORKING-STORAGE SECTION.
01  ZMAN      PIC S9(9)V9(6).
PROCEDURE DIVISION.
   move -0.2 to zman.
   add 1 to zman.
```

2.5.0 Sun Solaris distribution

- The Sun Solaris distribution could generate an “invalid path” error. This has been fixed.

2.5.0 Visual and Logical Hebrew

- The environment variable COB_RTL_CP must be defined to activate logical Hebrew handling,

even if the environment variable COB_CONSOLE_CP is set.

2.5.0 Invalid Referencing of Level-88

- The syntax: *IF fieldxxx = 1* where *fieldxxx* is a level 88 is now detected as invalid.

2.4

2.4.17a

New

2.4.17a round-fp

- The round-fp compiler flag controls the way COMP-1 or COMP-2 are “moved” into non COMP-1 or COMP-2 data items.
- If set to **yes**, the value is rounded to the number of decimals in the target field.
- If set to **no** (default) the value is truncated.

2.4.17a -save-temps

- The **-**save-temps compiler flag now creates a subdirectory named c where intermediate C files are stored.

2.4.17a line-seq-dos

- The line-seq-dos configuration file flag affects the END OF RECORD of LINE SEQUENTIAL files.
- If set to **yes** , records are terminated by CR/LF
- If set to **no** (default) records are terminated by LF

2.4.17a line-seq-unix

- The line-seq-unix configuration file flag is now obsolete.

2.4.16 Non-numeric RETURNING fields

- Non-numeric RETURNING fieldsare now supported.

2.4.16 ACCEPT FROM ESCAPE KEY

- ACCEPT FROM ESCAPE KEY is now supported.

2.4.16 DELETE FILE<sd filename>

- DELETE FILE <sd filename> is now supported. DELETE FILE <sd filename> erases the file and the optional index. The file must be closed.

2.4.16 Save/Load a list of breakpoints in Debugger

- Deet debugger has new options to save/load a list of breakpoints.

2.4.16 Change field values in Debugger

- Deet debugger enables change of string and numeric field value

2.4.15 odo-slide

- The odo-slide compiler flag affects data items that appear after a variable-length table in the same record; that is, after an item with an OCCURS DEPENDING clause, but not subordinate to it.
    - Data items described immediately following a table with an OCCURS DEPENDING clause can be subject to data loss, in a program with a “sliding” OCCURS DEPENDING ON clause;- that is, a program in which they immediately follow a table whose length is always being recalculated. This vulnerability emerges because their data address is constantly changing, as the size of the table changes.

- If set to **yes** (default), these items have variable addresses, and begin directly following the end of the table in its current state.
- If set to **no** these items have fixed addresses, and begin after the end of the space allocated for the table at its maximum length.

- WARNING: To be more close of the MF default:
  -std=mf and the "**mf.conf**" standard configuration now implies **odo-slide : no.**
- This is a change from behaviors of previous versions. If your code depends on odo-slide being set to yes, please add **odo-slide : yes** to your configuration file or add **–fodo-slide** to your compilation command line.

2.4.14 Gcc Builder for AIX Distribution

- There is now a gcc builder for the AIX distribution. This distribution is compiled and uses gcc as a support C compiler.

2.4.14 Improved Hebrew support

2.4.14 read-at-end-mf

- The read-at-end-mf configuration flag affects the compilation of the READ AT END and READ NOT AT END clauses on Indexed Files.
- If set to yes :When AT END and/or NOT AT END clause is defined in a READ statement and no NEXT or PREVIOUS is specified, a NEXT clause is implied.
- If set to no (default): The NEXT clause is not implied.

2.4.13 field of file-namesyntax

- The syntax *field* OF *file-name*is now supported.

```
FD TEST-FILE.
01 TEST-REC.
  02 XDATA     PIC X(4).

PROCEDURE       DIVISION.
   INITIALIZE XDATA OF TEST-FILE.
   …
```

2.4.13 Lib Optimizer

- The Lib Optimizer has had several improvements in code generation.

2.4.13 Initial support for Right To Left languages like Hebrew

- Define the environment variable COB_RTL_CP=<codepage> where <codepage> is the
- code page of the Right To Left Language. Then when displaying a string, the part of the string that must be written Right to left will be inverted.

  On Windows if you define the environment variable COB_CONSOLE_CP=<codepage> the codepage of the console is changed at program startup (This is equivalent to DOS command chcp<codepage>). Currently for Hebrew (1255, 8859-8, 862) are supported.

2.4.13 -fgcc-O-bug

- The compiler flag –fgcc-O-bug avoids incorrect code generation errors that have been seen with some versions of gcc when using the –O compiler flag.

2.4.13 Passing figurative constants as parameters in a CALL

- Passing figurative constants as parameters in a CALL is now supported.

- Example:

```
CALL “MYPROG” USING ZERO.
```

2.4.13 Empty WHEN Clauses in an EVALUATE Statement

- Using empty WHEN clauses in an EVALUATE Statement is now supported:
- Example:

```
EVALUATE condition
  …
  WHEN OTHER
END-EVALUATE
```

2.4.13 EXIT PROGRAM RETURNING x

- In the phrase:***EXIT PROGRAM RETURNING x***, the RETURNING clause is now optional

2.4.13 GOBACK RETURNING x

- GOBACK RETURNING x is now supported. The RETURNING clause is optional.

2.4.13 CALL RETURNING TO PIC X

- CALL RETURNING to PIC X is now supported

2.4.13 COB_NO_SIGNAL

- The runtime environment variable COB_NO_SIGNAL causes the runtime to not catch the signal which lets the system build a core dump. Setting COB_NO_SIGNAL can improve performance, while reducing the diagnostic capabilities of the runtime.

2.4.12 initialize-filler

- The configuration file flag initialize-filler configuration flag controls whether the INITIALIZE Statement initializes or does not initialize fields declared as FILLER.
- If set to yes
  - Fields declared as FILLER are initialized.
- If set to no (default)
  - Fields declared as FILLER are not initialized.

2.4.12 incomplete-occurs

- The configuration file flag incomplete-occurs affects the behavior of MOVEs to table items. Consider a data item declared as 01 TABLE OCCURS 10 PIC X.
- If set to yes
  - The phrase MOVE SPACE TO TABLE is equivalent to MOVE ALL SPACE TO TABLE.

2.4.12 complex-odo

- The compiler flag complex-odo is now obsolete. OCCURS DEPENDING ON is always considered to be complex.

2.4.12 Paragraph labels

- Paragraph labels no longer require a period.

2.4.12 Empty IF Statements

- Empty IF statements are now supported.

Example:

```
IF condition
ELSE
ENDIF
```

2.4.12 MOVE a TO b TO c TO d

- MOVE a TO b TO c TO d … is now supported.

2.4.12 Zero constant strings

- Zero constant strings are now supported . The formulation Z"My string" is stored with a binary 0 at the end.

2.4.11 Debugging preprocessed source code

- When using –preprocess and –g, Debugger (Deet) now skips code generated by the preprocessor and displays original source code lines.

2.4.11 SYMBOLIC CHARACTERS

- SYMBOLIC CHARACTERS are now supported.

2.4.11 SPECIAL-NAMES S01 through S05

- SPECIAL-NAMES S01 through S05 are now supported.

2.4.9 isam-extfh

2.4.9 isam-extfh-lib

- The configuration file flags isam-extfh and isam-extfh-lib enable the usage of EXTFH drivers for Indexed ISAM files.
- Usage for Indexed files:

```
isam-extfh:<DRIVER NAME>
isam-extfh-lib:<library to use for this extfh driver>
```

2.4.9 flat-extfh

2.4.9 flat-extfh-lib

- The configuration file flags flat-extfh and flat-extfh-lib enable the usage of EXTFH drivers for Sequential/Relative Files.
- Usage for Sequential/Relative files:

```
flat-extfh:<DRIVER NAME>
flat-extfh-lib:<library to use for this extfh driver>
```

2.4.9 Same name for the Select and Assign

- SELECT MyFile ASSIGN TO MyFile (Same name for the Select and the Assign) is now supported.

2.4.9 Redefined field in a USING Clause

- Usage of a redefined field in a USING clause is now allowed:

```
LINKAGE  SECTION.
```

```
    01 X          PIC X(4) VALUE 'ABCD'.
01 G          REDEFINES X.
      02 A         PIC X(2).
      02 B         PIC X(2).
    PROCEDURE DIVISION  USING G.
                …
```

2.4.9 UPPER-CASE and LOWER-CASE attributes in Screen Section

- In the SCREEN SECTION, the UPPER-CASE and LOWER-CASE attributes are allowed to force case conversion at input.

2.4.9 WHEN OTHER Clause preceded by other WHEN Clauses

- In EVALUATE clause, WHEN OTHER may be preceded by other WHEN Clauses

```
        evaluate wsKeyVal
                              …
          when 8
            display "8"
          when 9
            display "9"
          when 10
          when other
            display "other"
        end-evaluate
```

2.4.7 Support for Microsoft Visual Studio 2010

- Support for Microsoft Visual Studio 2010. Note that Visual Studio 2008 continues to be supported.

2.4.6 pack-comp-4

```
pack-comp-4: yes/no
```

The configuration file flag pack-comp-4 enables packing COMP-4 at minimum size needed whatever the value of “binary-size”

If set to yes

COMP-4 fields are stored as if binary-size=1—8

If set to no (default)

COMP-4 fields have default storage.

2.4.6 full-cancel

```
full-cancel: yes/no
```

The configuration file flag full-cancel provides a compiler flag to achieve the same effect as the setting of the environment variable COB_FULL_CANCEL.

Setting “full-cancel:yes” is equivalent to setting the environment variableCOB_FULL_CANCEL=1 at runtime.

2.4.5 COB_LOAD_PRIORITY

- The configuration file flag module-load-priority and the environment variable COB_LOAD_PRIORITY allow the user to change the default search order for a CALL’ed program.
- The configuration file flag module-load-priority allows for this change to be defined at compile time. The environment variable allows for this change to be defined at runtime.
- To resolve “myprog” in the phrase CALL “myprog”, the default behavior was first to look for the symbol “myprog” in the linked library. Then if the symbol was not found, the runtime would search for a shared library myprog.so (typically a COBOL module) where it would look into for a symbol myprog.

  This search order may now be reversed (first look for the shared library and if not found look into the linked symbols). This may be done at compilation or at runtime.

  To reverse the search order, a new configuration file flag has been created:

```
module-load-priority:yes
```

  The configuration file flag module-load-priority causes the search order change to be stored in the object file.

  Alternatively, the search order change can be determined at runtime by defining the environment variable :

```
export COB_LOAD_PRIORITY=1
```

2.4.4 quote

- The quote configuration file flag allow for the “quote” character to be configurable. The default setting for the “quote” character is “.
- To change the default value to a single quote, set the QUOTE configuration option as follows:

```
quote=””
```

2.4.4 Padding in variable-sized RECORD SEQUENTIAL records

- When handling RECORD SEQUENTIAL records written in variable size format (RECORDING V), READs and WRITEs performed by Micro Focus COBOL can include PADDING with the character 0 to bound a 4-byte position. COBOL-IT can now create the same PADDING allowance, with the creation of the configuration file flag variable-rec-pad-mf.

2.4.4 variable-rec-pad-mf

- The configuration fileflagvariable-rec-pad-mf lets you change the runtime behavior to read and write records that have been created with PADDING according to the Micro Focus convention.

```
variable-rec-pad-mf:yes
```

2.4.3 Switches

- 36 Switches are now available.

2.4.3 CitSORT improvements

- Better performances, and support for many new command like SUM, OUTFIL, EDIT, and CHANGE.

2.4.3 Improved performance of arithmetic computations

- Use of version 5.0.1 of GMP to improve performance of arithmetic computations

2.4.3 move-picx-to-pic9

- The configuration file flag move-picx-to-pic9 controls the way the runtime executes a MOVE from a PIC X to a PIC 9 USAGE DISPLAY data element.

```
move-picx-to-pic9=[cit | mvs | mf50]
```

Possible values are:

- cit : PIC 9 field if first initialized to all 0's, then the characters in the PIC X field are transferred.
- mvs : PIC X is copied as is into the PIC 9 field. Then, the last digit is checked and if it represents an EBCDIC sign character (A, B, …) the corresponding non signed value is replaced.
- mf50 : All PIC X data are copied to the PIC 9 field. This is identical to the MOVE of a PIC X field to another PIC X field.

2.4.2 –o [dir | filename]

- The compiler flag –o can take a directory name as a parameter. When the –o parameter is a directory name, the target object files are written to that directory.
- Usage: cobc -o [dir | filename]

2.4.2 –t [dir | filename]

- The compiler flag –t can take a directory name as a parameter. When the –t parameter is a directory name, the target list files are written to that directory.

  Usage: cobc -t [dir | filename]

2.4.0 USAGE NATIONAL

- USAGE NATIONAL is now supported.

Usage:

```
PIC N(10) USAGE NATIONAL
```

The NATIONAL characters are stored as UTF-16BE.

2.4.0 FUNCTION NATIONAL-OF

- Function NATIONAL-OF is now supported.
- Function NATIONAL-OF accepts an optional second parameter to specify the CODEPAGE. If not specified, the module CODEPAGE is used.

2.4.0 FUNCTION DISPLAY-OF

- Function DISPLAY-OF is now supported.
- Function DISPLAY-OF accepts an optional second parameter to specify the CODEPAGE. If not specified, the module CODEPAGE is used.

2.4.0 FUNCTION LENGTH

- Function LENGTH, when applied on data items described as USAGE NATIONAL returnsthe number of NATIONAL (UTF-16) CHARACTERS.

2.4.0 FUNCTION BYTE

- Function BYTE-OF returns Memory size.

2.4.0 CONVERSIONS ON MOVE FROM/TO USAGE DISPLAY

- MOVEs from and to USAGE DISPLAY data items do data conversion using the module CODEPAGE.

2.4.0 -codepage

- The –codepage compiler flag defines the code page used.

```
-codepage “code-page”
```

2.4.0 -utf-8

- The –utf-8 compiler flag causes the compiler to accept source code stored in UTF-8 format, and allows identifiersto be non-ascii.
- Usage:

```
-utf-8
```

Fixes

2.4.17a MOVE of numeric field to JUSTIFIED field

- MOVE of numeric field to JUSTIFIED field did not apply the JUSTIFIED attribute to the numeric field. This has been fixed.

2.4.16 Occurs Depending phrases with variable declared in Linkage

- When an OCCURS DEPENDING in WORKING STORAGE describes the “depending” field in the LINKAGE SECTION, and the –g compiler flag was used, the generated code did not compile at the C level. This problem has been fixed.

2.4.14 Sequential and Relative OPTIONAL files with OPEN I-O

- Sequential and Relative OPTIONAL files opened for I-O are now created if they do not exist.

2.4.14 AIX Warning on Duplicated Symbol

- On AIX, a duplicate symbol did not always generate a warning when linking the executable. This has been fixed.

2.4.12 INITIALIZE FILLER behavior

- INITIALIZE FILLER behavior fixed:

  When FILLER was defined in a record, the INITIALIZE statement was applied to the FILLER. This behavior has been corrected. Now, the FILLER items are left unchanged by the INITIALIZE statement unless you use INITIALIZE WITH FILLER.

2.4.12 INITIALIZE-TO-VALUE

- The Configuration flag: 'initalize-to-value' has been renamed to 'initialize-to-value'.

2.4.9 C optimization with –O Compiler flag

- C optimization was not correctly requested when -O flag was used. This has been fixed.

2.4.9 Space in COPY ... REPLACING phrase

- In COPY … REPLACING == :TAG: == BY == other ==
  Spaces between 'other' and == are now ignored.

2.4.9 Using the –E and –t compiler flags with free format source code

- Preprocessing (-E or –t) of free format source no longer adds 8KB of new-linesat the end of the file.

2.4.9 UNSTRING in multi-threaded runtime environment

- UNSTRING in multithreaded context has been fixed

2.4.9 Ambiguous compiler error

- Source like :

```
WORKING-STORAGE SECTION.
01 G1.
02 X          PIC X VALUE "X".
02 G1.
03 Y        PIC X VALUE "Y".
PROCEDURE       DIVISION.
        DISPLAY G1 NO ADVANCING
```

  no longer generates an 'ambiguous; need qualification' compiler error.

2.4.8 a VBISAM performance degradation with variable-sized records

- When using variable-sized records, the performance of the file system would degrade dramatically as the number of records increased. This has been fixed.

2.4.8 a VBCheck improvements

- VBCheck, the file check and repair utility of VBISAM, has been made stronger and safer.

2.4.8 a SIZE ERROR check on DEPENDING ON variable

- OCCURS a TO b DEPENDING ON xx. When checking for a size error condition on xx, the value of xx was not checked against the limit fixed by the a TO b clause. This has been fixed.

2.4.8 a Complex OCCURS … DEPENDING ON Clauses

- Under certain circumstances, complex OCCURS … DEPENDING ON .. clauses in file records were not correctly computed. This has been fixed.

2.4.8 a Division by 0 in optimized code

- When using the -O or -fbin-opt compilation flags, the division by zero was not checked and

a system error could result, which could create a crash in the program. This has been fixed. The division by zero exception is now detected, and a COBOL exception is raised.

2.4.7 Thread-safe runtime

- A defect was fixed in the thread-safe runtime. This fix impacts only programs running in a multi-threaded environment.

2.4.6 Filenames with spaces in their pathname

- Filename with white space in the path are now enclosed into " " when transmited to the C compiler

2.4.6 Rare and Minor compilation crashes

- A number of rare and minor compilation crashes have been fixed.

2.4.6 EXTFH with external-mapping configuration file flag

- When using EXTFH and the configuration flag :

```
external-mapping: yes
```

The filename for files declared as EXTERNAL is no longer converted by the COBOL-IT runtime. The flag bit number 4 of the "flags" field of the FCD structure (at offset 25) is now correctly set.

2.4.5 Group items with multiple OCCURS DEPENDING ON Clauses

- Complex OCCURS DEPENDING ON was mis-computed in group items containing multiple OCCURS DEPENDING clauses. This has been fixed.

Example:

```
02 AAA.
        05 X  PIC 99.
        05 Y  PIC 99 COMP.
        05 Z  PIC 99 COMP-3.
        05 XX  OCCURS 5 DEPENDING ON X.
          10 XXX PIC X(1).
        05 YY  OCCURS 5 DEPENDING ON Y.
          10 YYY PIC X(2).
        05 ZZ  OCCURS 5 DEPENDING ON Z.
          10 ZZZ PIC X(3).
```

2.4.5 No Profiling Output generated

* When using profiling, programs terminated by STOP RUN did not output the profiling information. This has been fixed.

2.4.4 Non signed USAGE DISPLAY (PIC 9(..)) comparison

- The comparison below could produce an incorrect result where VALH was declared as PIC 9(8) or PIC 9(9). This has been fixed.

```
01 VALH PIC 9(8) .

MOVE HIGHVALUE TO VALH.
IF VALH > ”12345678” …
```

2.4.3 VBISAM bug when 2 files open at the same time

- A VBISAM bug was detected when 2 files were opened at the same time. This has been fixed.

2.4.3 Code cleaning

- Some code cleaning was done

2.4.2 Record locking error

- An error in VBISAM record locking has been fixed.

2.4.2 Level-01 alignment on 64-bit systems

- On 64-bit systems, the default level 01 alignment is now 8 bytes

2.4.0 EXTFH interface supports LINE ADVANCING syntax

- The LINE ADVANCING clause is now transmitted to the EXTFH driver.

2.4.0 Computations with COMP-2 and COMP-1 data items

- COMP-2 and COMP-1 are now computed as true floating point data items.

1.4.0 COPY/REPLACE fix

- COPY/REPLACE did not work correctly in some cases.

  This has been corrected.

2.3

2.3.1

New

2.3.1 Thread-safe support enhancement

- Thread safe internal structures now use the pthread library, which is available in AIX, Linux, HPUX Itanium, Sun Solaris and Windows operating environments.

Fixes

2.3.1 DISPLAY and START fixes

- There have been some minor fixes in the DISPLAY and START verbs

2.3.0 Anomalies associated with usage of –Os compiler flag

- Usage of the -Os Optimizer could introduce some random behaviors.

  This has been corrected. Any modules compiled in a previous release with the –Os compiler flag must be recompiled.

2.3.0 Parser fix

- Minor parsing defects were detected.

  These have been corrected.

2.2

2.2.0

New

2.2.0 Runtime "abort" status codes

- Runtime status codes returned when the runtime aborts:

128	CALL Unresolved
129	NULL name parameter passed to 'cobcancel'
130	'cobcall' - Runtime has not been initialized
131	Invalid number of arguments to 'cobcall'
132	NULL name parameter passed to 'cobcall'
133	'cobfunc' - Runtime has not been initialized
134	NULL name parameter passed to 'cobsavenv'
135	NULL name parameter passed to 'coblongjmp'
136	Cannot acquire %d bytes of memory – Aborting
137	BASED/LINKAGE item '%s' has NULL address
138	'%s' not numeric: '%s'"
139	OCCURS DEPENDING ON '%s' out of bounds: %d
140	Subscript of '%s' out of bounds: %d
141	Offset of '%s' out of bounds: %d
142	Length of '%s' out of bounds: %d
143	EXTERNAL item '%s' has size > %d
144	'cobcommandline' - Runtime has not been initialized
145	Parameter to SYSTEM call is larger than 8192 characters
146	Invalid context file %s for reading
147	Can't open context file %s for reading
148	Can't write context file %s key
149	Can't open context file %s for writing
150	Unexpected context mode %d
151	Context file not open for read/write
152	Can't read context file %s data
153	Runtime is unable to acquire temporary file
154	extfh_indexed_open : invalid open mode %d for %s
155	extfh_indexed_start : invalid condition
156	extfh_read : invalid read option %d
157	extsm/SORT : maximum USING/GIVING clauses exceeded
158	extfh/extsm Internal error

159	Failed to initialize curses
160	cob_init() has not been called
161	Code generation error - Please report this to support@cobol-it.com
162	ERROR - Recursive call of chained program
163	Stack overflow, possible PERFORM depth exceeded
164	CBL_xxx CALL Insufficient parameters count
253	Function only available in enterprise version
254	Fatal error see messages
255	Undefined error

2.2.0 EXTFH and EXTSM interfaces in 64-bit mode

- COBOL-IT's EXTFH and EXTSM interfaces now use the FCD3 in 64-bit mode with compatibility with the standard used by MF.This enables usage of EXTFH/EXTSM with 64 bits interfaces.FCD2 is still used for 32 bits mode. SyncSort'sDMExpresshas been validated with the new interfaces. Any programs using EXTFH in 64-bit mode must be recompiled.

Fixes

2.2.0 cobmf aborts when it can't create a configuration file

- cobmf (The MF command line emulator) would abort if it could not create a configuration file in the directory of the first file passed as an argument.

  This has been corrected. Now, if cobmf fails to create a configuration file in that directory, it will create the configuration file in the /tmpdirectory, or the directory named by the TMP environment variable.

2.2.0 Command-line parsing

- Command-line parsing could have different behavior depending on the execution platform.

  This has been corrected. The code has been unified to provide the same behavior on all platforms.

2.1

2.1.1

New

2.1.1 –fC-cmd-line

- The compiler flag -fC-cmd-line, when used with –x, causes the program to receive command line parameters as though they were given in C

2.1.1 Tally Register

- The predefined register TALLY has been added. It is defined as:

```
01 TALLY GLOBAL PICTURE 9(9) USAGE COMP-5 VALUE ZERO.
```

2.1.1 tally-register

- The configuration file flag tally-register, when set to NO, disables the creaton of the predefined register TALLY.
- Usage:

```
tally-register:no
```

2.1.1 CALL "myCfunction"... RETURNING INTO field-structure

- If myCFunction returns a pointer to a data structure, the contents of the structure is copied into the given field .

2.1.1 CALL "myCfunction" ... RETURNING ADDRESS OF field

The pointer returned by "myCfunction" is used as the ADDRESS of *field*. *Field* must be declared BASED.

2.1.0 Lib Optimizer

- The Lib Optimizer optimizes the generation of C code by the COBOL-IT compiler. The Lib Optimizer splits large modules into separate C functions. The C compiler can then more effectively optimize the resulting code.

2.1.0 -Os

- The –Os compiler flag causes large modules to be split, and optimized C code to be produced.

2.1.0 –Os -O

- The –Os –O sequence of compiler flags causes the C compiler to optimize to reduce object size when compiling the optimized C code.

2.1.0 –O –Os

- The –O –Os sequence of compiler flags causes the C compiler to optimize to maximize execution speed, when compiling the optimized C code. Note that objects optimized for performance may be larger in size.

2.1.0 –O

- The –O compiler flag causes optimized C code to be produced, and causes the C compiler to optimize to maximize execution speed. However, with very large modules, some C compilers may fail. The –Os compiler flag can be used to break the large modules down, and avoid this problem.

2.1.0 Cobmf -CIT

- The –CIT command line option is used to pass additional commands to the COBOL-IT compiler when using cobmf command line emulator.

  Usage :

```
cobmf -CIT "-v -Os -O" myprog.cob
```

2.1.0 COB_OPTSIZE_FLAG

- The environment variable COB_OPTSIZE_FLAG is used to instruct the C compiler to reduce code size. COB_OPTSIZE_FLAG was formerly named COB_OPTSIZE_FLAGS.

2.1.0 Enable runtime exception checking

- Runtime exception checking is enabled with the support of the following compiler configuration flags.
  - EC-BOUND-PTR
  - EC-I-O
  - EC-BOUND-SUBSCRIPT
  - EC-BOUND-REF-MOD
  - EC-DATA-INCOMPATIBLE
  - To enable the specific exception check, set the associated compiler configuration flag to yes in the compiler configuration file.

    Usage:

```
#Runtime Exception Check
#set to yes to enable runtime exception

#Bound Pointer and base vars
EC-BOUND-PTR:no
#I-O
EC-I-O:yes

#Field Subscript check Field(idx)
EC-BOUND-SUBSCRIPT:no

#Field Ref check        Field(offset:len)
EC-BOUND-REF-MOD:no

#Decimal expand When moving/computing to numeric field, check
#if source field hold a valid numeric value
EC-DATA-INCOMPATIBLE:no
```

Fixes

2.1.1 Handling errors in the formulation of the compile command

- If the executable (-x) was requested and the main module was given as an object file (.o) instead of as a source file (.cob), then the program received NULL as a command line parameter.

  This has been corrected.

2.1.0 Memory Leak introduced in version 2.0.x

- **VERY IMPORTANT :**
  A memory leak introduced in version 2.0.x runtime has been corrected in this version.
  All programs MUST BE RECOMPILED to be compatible with Version 2.1.0 runtime

2.0

2.0.10

New

2.0.10 -Os

- The -Os compiler flag optimizes C generated code to minimize object size.

2.0.10 COB_OPTSIZE_FLAGS

- The environment variable COB_OPTSIZE_FLAGS names flags to be used in the C compilation phase.
- Usage: To cause the –Os flag to be used by the C compiler

```
export COB_OPTSIZE_FLAGS=-Os
```

2.0.10 -foptional-file

- The –foptional-file compiler flag causes all SELECT statements that do not specify OPTIONAL or NOT OPTIONAL to be considered OPTIONAL.
- If the compiler flag is not present or if -fno-optional-file is specified, NOT OPTIONAL is the default value.
- The implementation of –foptional-file is designed to be compatible the MF –C OPTIONAL-FILE flag.

Fixes

2.0.10 OCCURS DEPENDING ON variablein FILE SECTION

- An OCCURS DEPENDING ON *variable* clause in the FILE SECTION that referred to a *variable* declared in the Working-Storage Section would cause an error. This has been fixed.

2.0.8b

New

2.0.8 b COBOPT

- The COBOPT environment variable is supported by cobmf. The COBOPT environment variable stores command-line compiler flags, for use with cobmf.

2.0.8 b COBITOPT

- The COBITOPT environment variable stores command-line compiler flags, for use with cobc. COBITOPT is designed to be compatible with the MF environment variable COBOPT.

2.0.8 b COPY/REPLACING supports LEADING/TRAILING syntax

- The terms LEADING and TRAILING are now supported in the COPY/REPLACING syntax.
- Usage:

```
COPY … REPLACING LEADING ==word== BY ==word==
COPY … REPLACING TRAILING ==word== BY ==word==
```

2.0.8 b cobsetjmp.h added to include directory

- To increase MF compatibility, cobsetjmp.hhas been added to the include directory

2.0.8 b –ftruncate-listing

- The compiler flag -ftrucate-listing causes comments generated into the listing file generated by the –t compiler flag to be truncated to 76 characters.

2.0.8 b Runtime Exit Procedure

- A runtime exit procedure has been added that is called by the runtime when exiting just before terminating the process. This may be used by a monitoring system to catch a runtime abort or to exit and take control again .

```
typedef void (*cob_rtd_exit_proc) (void *, int );
void  cob_set_exit_rtd_proc (COB_RTD,
                                        cob_rtd_exit_proccob_exit_proc);
```

2.0.8 b ADD/SUBTRACT Performance Enhancements

- Improved speed for ADD and SUBTRACT of integer value (field or constant)to/from USAGE DISPLAY field.

2.0.8 b -f77-opt

- The –f77-opt compilation flag that aggressively optimizes the use of integers stored in USAGE DISPLAY or PACKED fields in level-77 data items.

2.0.6 Thread memory release and support for a program monitor

- Added Thread memory release and support for a program monitor taking control of the abort & exit of the COBOL-IT runtime. A typical “program monitor” should use the following scheme to call cobol programs:

```
/* Allocate Runtime Data memory for this thread */
cit_runtime_t * constrtd = cob_get_rtd();

int (*func)();
int res;

/*initialize Cobol Thread runtime */
cob_init(rtd, argc, argv);

/* Prepare the Runtime to exit with a long jump instead of exit(x)*/
res = cob_setjmp(rtd);
if ( res == 0) {
        /* Find the COBOL Program */
        func = cob_resolve(rtd, "prog");
        if ( func ) {
                /* Call it and all other needed*/
                func();
        }
        /* if the Program goes Here the the COBOL exit normally */
        /* with a GO BACK                                       */
        …

} else if ( res > 0 && res < 127) {
        /* COBOL Program exit normally through STOP RUN */
        /* res hold the RETURN-CODE */
        …

} else {
        /* COBOL Program exit on runtime abort */
        /* res hold the error code */
        …

}

/* Before leaving the Thread Release Runtime Data */
cob_rtd_tidy ();
```

2.0.4 *Symbol in object file for PROGRAM-ID*

- The symbol in the object file for the PROGAM-ID and ENTRY names are now generated as given in the source code, and are generated in upper-case, for compatibility with MF.

2.0.4 *Search for symbol to resolve CALL statement enhanced*

- When CALL statements try to find a symbol in a .so, if the exact symbol is not found, the upper-cased symbol is searched.

2.0.4 *PERFORM UNTIL EXIT*

- PERFORM UNTIL EXIT is now supported

2.0.3 *Reducing code size*

- Reduced code size when building non thread-safe program (-fthread-safe not used)

2.0.3 external-link

- The compiler configuration flag external-link can be set to a function name, to cause that function name to be declared as an external non-COBOL symbol.
- Usage:

```
external-link: functionName
```

- Using the compiler configuration flag external-link will cause the code "CALL 'function-name' " to generate more efficient code.

2.0.3 oraclex.symb, tuxedox.symb

- The compiler can be provided with lists of the Oracle and Tuxedo symbols.
- Usage:

```
cobc -conf=+oraclex.symb
cobc -conf=+tuxedox.symb
```

2.02 64-bit CitSORT

- CitSORT is available in 64-bit distributions

2.02 COBOL-IT Preprocessor now ignores INCLUDE statements

When using -preprocess=cmd the COBOL-IT Preprocessor now ignores INCLUDE statements even if not enclosed in an EXEC END-EXEC sequence.Those INCLUDE statements must be processed by the external preprocessor.

2.02 Reducing object size produced with –O or –fbin-opt

- When using -O or -fbin-opt object sizes are reduced by about 30% on very big modules.

2.0.1 -preprocess=cmd

- The –preprocess=cmd compiler flag implements the COBOL-IT integrated pre-processor. Preprocessors (like Oracle procob) may now be called by the compiler after the COBOL preprocessing of COPY/REPLACING clauses has been completed.
- Usage: -preprocess=cmd
  - cmd is the external preprocessor command that take 2 parameters :The first parameter is the input file name. The second parameter is the output file name. Usually a script is used. A sample script calling Oracle Pro*COB is located in $COBOLITDIR/bin/citprocob.sh, which executes the command:
  - procob format=TERMINAL iname=$1 oname=$2

2.0.1 -fthread-safe

- The -fthread-safe compiler flag causes thread-safe code to be generated.

2.0.1 Thread-safe code generation and runtime

- The COBOL-IT runtime is now thread safe.

2.0.1 Support for COMP-6 data.

- COMP-6 data is now fully supported.

2.0.1 accept-but-ignore-comp6-sign

- If accept-but-ignore-comp6-sign is set to yes, the sign declaration is accepted by ignored.

• accept-but-ignore-comp6-sign:[yes|no] default no

2.0.1 signed-comp6-as-comp3

- If signed-comp6-as-comp3 is set to yes, the sign declaration is accepted and the field is considered as a COMP-3.
- Usage:

• signed-comp6-as-comp3:[yes|no] default no

2.0.1 –fdebug-exec

- The –fdebug-exec compiler flag affects the tracing of Exec statements when debugging. When using the Integrated Preprocessor Interface, the default behavior of the debugger is to –not- trace (display) the code generated by the external preprocessor. Only the original source EXEC statementsare shown. The –fdebug-exec compiler flag enables the tracing (debugging) of the generated code.

2.0.1 COB_FULL_CANCEL

- The COB_FULL_CANCEL environment variable, when set to 1, forces the unloading of code and data when a CANCEL verb is executed. Note that by default, the runtime does not unload the module and data, in favor of faster performance.

2.0.1 –constant "key=value"

- The –constant "key=value" compiler flag provides a way to define constants that can be tested for purposes of conditional compilation. After using the –constant "key=value" compiler flag, the conditional compilation below will test true.
  - $if key=value
  - $else
    $end

2.0.1 ALTER statement

- The ALTER statement is now supported for legacy purposes. Programmers are urged to not use this construction and replace it in existing programs.

2.0.1 Generated C code simplified

- Generated C code has been simplified. This does help compiling very large modules on platform like AIX. No runtime performance improvement should be expected.

2.0.1 Time to reduce C code greatly reduced

- The time needed to generate the C code (formally compile the COBOL) has been heavily reduced for large modules (up to a factor of 9 on very large modules)

Fixes

2.0.8 b –E output

- The -E compiler flag preprocesses COBOL code, without doing any translation to "C", compilation, assembly, or linking. The preprocessed code is output to stdout by default, and reproduces the preprocessed COBOL source code. Output from the –E compiler flag now respects the original source mode (free or standard).

2.0.8 b Thread-safe interface

- cob_set_rtd () and cob_rtd_tidy() functionshave been added to the thread-safe interface

2.0.8 b Sign digit incorrectly set on MOVE of COMP-3 to COMP-6

- When MOVEing a non signed COMP-3 or a COMP-6 to a signed COMP-3 the sign digit was not set correctly. This has been fixed.

2.0.6 not-reserved

- The not-reserved configuration file flag did not allow the reserved word INCLUDE to be ignored. This has been fixed.

```
not-reserved:INCLUDE
```

2.0.6 Free Format supports comments in the IDENTIFICATION DIVISION.

- In Free format the IDENTIFICATION DIVISION comment like DATE-COMPILED are now supported again.

2.0.4 Various fixes to the Integrated Preprocessor interface

- Fixes to the Integrated Preprocessor Interface :
  - Line continuation characters (- in column 7) are supported.
  - Commas “,” and semi-colons “;” are not removed before external preprocessing
  - The line length of generated comments is limited to 100 characters.

2.0.4 Use of ZERO figurative constant in IF condition

- Using the figurative constant ZERO in an IF condition could cause errors. This has been fixed.
- Example:

```
IF W-VAR1 = W-VAR2 AND NOT ZERO
```

2.0.4 Complex IF constructions

- Certain complex IF constructions could be interpreted incorrectly. This has been fixed.
- Example:

```
IF W-VAR1 > 0 AND ( < 3 OR  > 5 )
```

2.0.4 DEET debugger running with large fonts

On some systems, using large fonts in the Deet debugger could cause problems. This has been fixed.

2.0.3 –fgcc-bug

- On very large modules, the C compilation phase may enter an infinite loop when using GCC. The -fgcc-bug compiler flag fixes the problem.

1.2

1.2.20

New

1.2.20 EXIT PROGRAM RETURNING x

- EXIT PROGRAM RETURNING x is now supported as alias of

```
MOVE x TO RETURN-CODE
EXIT PROGRAM
```

1.2.20 –g produces smaller debug object files

- -g no longer generates C debugging information. Only Cobol (deet) debugging information is generated. This was done to reduce the size of the debug object.

1.2.20 –G generates C-level debugging information

- Users wishing to have C-level debugging information in their debug objects must compile with -G.

1.2.20 Cobmf improvements

- The MF command line emulator has been improved.

1.2.16 WRITE myfile FROM FUNCTION

- WRITE myfile FROM FUNCTION ... is now supported

1.2.16 Special-Names CSP

- Special name CSP is now supported

1.2.16 EXTERNAL and GLOBAL attributes not mutually exclusive

- A data item may be defined IS EXTERNAL and IS GLOBAL simultaneously.

1.2.13 zLinux port

- COBOL-IT has been ported to the zLinux platform.

1.2.13 Performance improvements on Linux x86 platforms

- On Linux x86 and x86_64, the arithmetic library has been improved and selects at runtime the best code optimized for the running CPU. (GMP 4.3.1)

1.2.12 Improved IBM Source level support

- A period "." at the end of a COPY statement is not required.
- ++COPY and ++INCLUDE are considered synonyms of COPY

1.2.12 Improved listing (-t) output

- Comment are now conserve and output is reusable to be compiled

1.2.12 NEW COBOL verb "CHECKPOINT"

- CHECKPOINT saves the current status of a program (Field Value, call stack , perform stack) into a file named <checkpoint prefix><program id>.ctx.
- When used with CONTINUE, the runtime saves the status and continues execution.
- When used with EXIT RETURNING *value*, the runtime saves the status, and exits the

program (and all calling parent programs) returning the value.

- When called back with checkpoint file, the runtime reloads the program status and continues the execution at the statement just following the CHECKPOINT line. If a GIVING clause is specified, the field is set to "1" when reloading and to "0" when execution is continuing.
- Usage:

```
CHECKPOINT [checkpoint prefix] CONTINUE [GIVING field]
or
CHECKPOINT [checkpoint prefix] EXIT [RETURNING|WITH] [return value] [GIVING field]
```

Reloading a checkpoint is done by using the –reload compiler flag, or by using the –reload compiler flag together with the –checkpoint compiler flag, and the <checkpoint prefix>:

```
cobc –-reload myprog
or
cobc -–reload -–checkpoint <checkpoint prefix>myprog
```

1.2.12 -fcheckpoint

The –fcheckpoint compiler flag enables the use of checkpoint processing. Where checkpoint processing is being used, all modules, and related sub-programs need to be compiled with –fcheckpoint.

1.2.11 Increased compatibility with MF

- SEQUENTIAL File with variable record length (RECORDING MODE "V") are now compatible with MF (read and write)
- CitSORT adapted to support MF variable length recording mode.

1.2.10d Improvide compatibility with MF

- The SCREEN SECTION has been improved to provide more compatibility with MF.
- Compatibility with MF reading of parameters, as described below, has been improved.

```
LINKAGESECTION.
01  ZONECHANGE.
  03    CMD-LENGTH       PICTURES9(4)  USAGEBINARY.
  03    CMD-LINE.
05  CMD-CHAR         PICTURE X
OCCURS1TO99DEPENDINGON CMD-LENGTH.

PROCEDUREDIVISIONUSING  ZONECHANGE
```

Fixes

1.2.19 MOVE of PIC 9(n) to PIC X(n) while using SPZERO

- When using the MF compatibility option SPZERO, a MOVE of PIC 9(n) to PIC X(n) could produce incorrect results. This has been fixed.

1.2.19 VBISAM error 61 (file sharing error)

- VBISAM could produce an error 61 (File sharing error) when 2 processesOPENedINPUT the same INDEXED file at the same time. This has been fixed.

1.2.17 END-PERFORM not mandatory.

- END-PERFORM is not mandatory any more. The PERFORM Statement may be terminated by a period, as in the code below.

```
PERFORM VARYING IND-VAR  FROM 8 BY -1
UNTIL WS-APPO(IND-VAR) NOT = SPACES.
```

1.2.15 Comments in COPY/REPLACING

- Comments inside a COPY REPLACING list was not supported. This has been fixed.

1.2.15 Compilation Error when using DECIMAL-POINT IS COMMA

- The code below would incorrectly generate a compilation error when using decimal-point is comma. The compiler would return the error “Y requires 3 subscripts.”. This has been fixed.

```
     WORKING-STORAGE  SECTION.
01 G1.
 02 X          OCCURS 3.
 03 X         OCCURS 3.
04 Y        PIC 999 OCCURS 3.
01 D1            PIC 99 VALUES 1.
 ...
 move 122 TO Y(1,2,D1)
```

1.2.13 Fixes to the AIX build

1.2.13 Fixes to the RedHat 4.x build

- Linux distribution is now compatible and tested on :
  - RedHat : RHEL 4.x & RHEL 5.x
  - Suse : SLES 10 & 11 , OpenSuse 10.3

1.2.13 Variable-length VBISAM records with GCC 3.4

- Aliasing in VBISAM made variable record size VBISAM files non-working with GCC 3.4. This has been fixed.

Debugger fixes

- In Windows operating environments, if an invalid program name is passed to the debugger at startup , Wish85 would loop. This has been fixed.
- If you accidentally close the COBCRUN window Wish85 would loop. This has been fixed.

1.2.11 START behavior

- A Bug fix in indexed files (introduced in 1.2.10) did not work as expected. (see code

below.). This has been fixed.

```
START  IX-FS1
KEYISNOTLESSTHAN R-RECKEY-1-7.
READ    IX-FS1
```

1.2.11 TUXEDO on HP-UX

- Support for TUXEDO on HP-UX has been fixed

1.2.11 IS EXTERNAL didn't share field between programs

- A field declared as EXTERNAL Field was not being shared between programs. This has been fixed.

1.2.10d Bug fix in indexed files

```
START ... KEYISEQUAL XXX
INVALIDKEY
...
NOTINVALIDKEY
          ...
END-START
```

In some case neither "INVALID KEY" clause or "NOT INVALID KEY" clause was called

1.2.10d Fix code generation with length function

```
move 2 to b(1).
display "length(a(1:b(1)))= " function length(a(1:b(1))).
```

was returning -1 instead of 2.

1.2.10d exit-program-forced

- The exit-program-forced configuration file flag changes the way that the EXIT PROGRAM statement is handled.

  If set to no (default) the program is exited only if it is not the main program.
  If set to yes the EXIT PROGRAM verb always exits the current program.