FUJITSU

Fujitsu Software NetCOBOL V13.0

User's Guide (Third-Party COBOL Resource Migration)

Windows(64)/Linux(64)

J2UL-2880-01ENZ0(00)
July 2023

Preface

This manual describes the function to use when migrating third-party COBOL resources to NetCOBOL. Included are notes and reference information.

Trademarks

- Microsoft, Windows, and Windows Server are trademarks of the Microsoft group of companies.
- Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.
- Red Hat and Red Hat Enterprise Linux are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.
- Intel and Itanium are trademarks of Intel Corporation or its subsidiaries.
- Micro Focus is a trademark or registered trademark of Micro Focus or its subsidiaries or affiliated companies in the United Kingdom, United States and other countries.
- UNIX is a registered trademark of The Open Group.
- All other trademarks are the property of their respective owners.

Abbreviations

The following abbreviations are used in this manual:

Product name	Abbreviation
Microsoft(R) Windows Server(R) 2022 Datacenter Microsoft(R) Windows Server(R) 2022 Standard	Windows Server 2022
Microsoft(R) Windows Server(R) 2019 Datacenter Microsoft(R) Windows Server(R) 2019 Standard	Windows Server 2019
Microsoft(R) Windows Server(R) 2016 Datacenter Microsoft(R) Windows Server(R) 2016 Standard	Windows Server 2016
Microsoft(R) Windows Server(R) 2012 R2 Datacenter Microsoft(R) Windows Server(R) 2012 R2 Standard Microsoft(R) Windows Server(R) 2012 R2 Foundation	Windows Server 2012 R2
Microsoft(R) Windows Server(R) 2012 Datacenter Microsoft(R) Windows Server(R) 2012 Standard Microsoft(R) Windows Server(R) 2012 Foundation	Windows Server 2012
Windows(R) 11 Home Windows(R) 11 Pro Windows(R) 11 Enterprise Windows(R) 11 Education	Windows 11 or Windows 11 (x64)
Windows(R) 10 Education Windows(R) 10 Home Windows(R) 10 Pro Windows(R) 10 Enterprise	Windows 10 or Windows 10 (x64)
Red Hat(R) Enterprise Linux(R) 9 (for Intel64) Red Hat(R) Enterprise Linux(R) 8 (for Intel64)	Linux(64)
Red Hat(R) Enterprise Linux(R)	Red Hat Enterprise Linux

Product name	Abbreviation
COBOL product of Micro Focus	Micro Focus COBOL

In this manual, when all the following products are indicated, it is written as "Windows(x64)" or "Windows."

- Windows Server 2022
- Windows Server 2019
- Windows Server 2016
- Windows Server 2012 R2
- Windows Server 2012
- Windows 11(x64)
- Windows 10(x64)

Purpose of This Manual

This manual explains how to migrate development resources such as third-party COBOL source programs and COBOL files to NetCOBOL.

Intended Readers

This manual is intended for those who are migrating development resources such as third-party COBOL source programs and COBOL files to NetCOBOL.

Prerequisite Knowledge

Readers are required to have the following knowledge to read this manual.

- Basic knowledge of COBOL syntax

Structure of This Manual

The structure of this manual is as follows:

Chapter 1 Overview of COBOL Resource Migration

Explains the overview of resource migration.

Chapter 2 Migrating COBOL Source Program and COBOL Libraries

Explains the pre-compiler source conversion function that is used for migration of COBOL source program and COBOL libraries.

Chapter 3 Migration COBOL File

Explains the COBOL file migration tool that is used for migrating COBOL files.

Appendix A Migration Reference (from Micro Focus COBOL)

Explains requirements for the pre-compiler convert function and requirements for the information file when converting a COBOL source program and COBOL library from Micro Focus to NetCOBOL.

Appendix B Conversion Result File

Explains the file written by the pre-compiler source conversion function.

Appendix C How to Customize Reserved Words

Explains how to customize reserved words in pre-compiler source conversion function.

Appendix D Restrictions

Explains the limitations when using pre-compiler source conversion function.

System-specific Functions

Some parts of the COBOL common syntax described in this manual depend on system functions, and differ among systems.

Such parts are indicated by the following system names:

Indicator	Corresponding system	Corresponding product
[Linux64] or 	Red Hat Enterprise Linux 9 (for Intel64) Red Hat Enterprise Linux 8 (for Intel64)	NetCOBOL (64bit) V13
[Winx64] or 	Windows Server 2022 (*1) Windows Server 2019 (*2) Windows Server 2016 Windows Server 2012 R2 Windows Server 2012 Windows 11 (x64) (*1) Windows 10 (x64)	NetCOBOL (64bit) V12 NetCOBOL (64bit) V12a

*1: Windows Server 2022 and Windows 11 are supported on NetCOBOL V12a or later.

*2: Windows Server 2019 is supported on NetCOBOL V12.2 or later.

Export Controls

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Other Notes

- For [Winx64], replace "directory" with "folder".
- For [Winx64], replace ".o" with ".obj".

July 2023

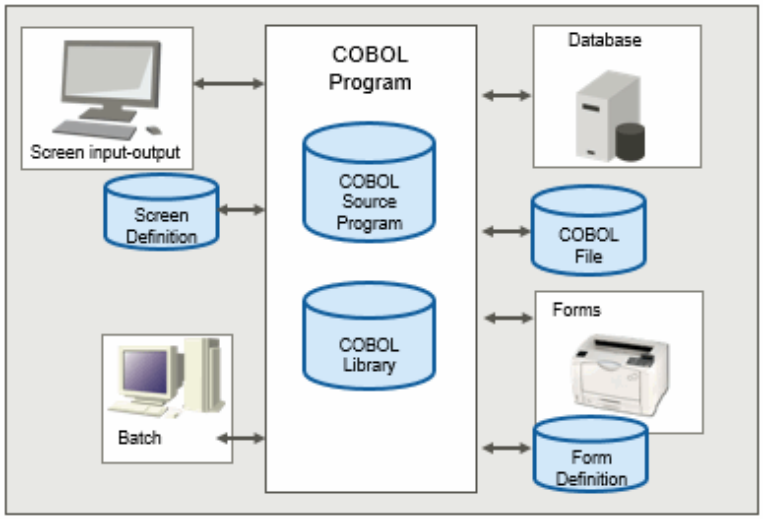
Contents

Chapter 1 Overview of COBOL Resource Migration

COBOL (COmmon Business Oriented Language) is a programming language designed for business use. Using this language, it is possible to easily read and understand programs written in a language that is similar to English, and be able to express specific business requirements like decimal operation, file processing, forms creation etc.

Each COBOL product is provided with a development environment and an execution environment for executing applications.

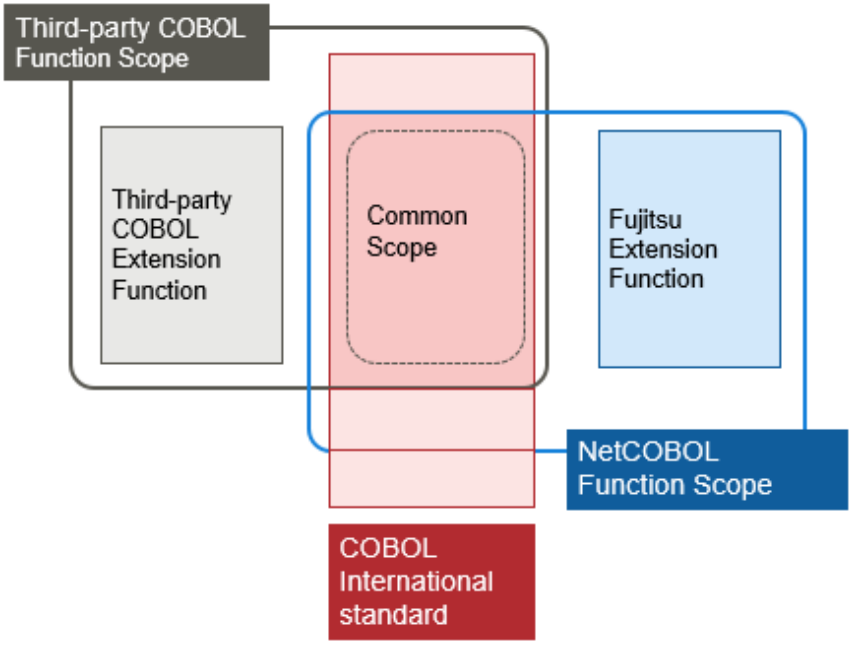
COBOL applications are composed from COBOL programs created based on COBOL international standards, various database, screen definition, and form definition etc. The following are called COBOL resources and include COBOL source program, libraries, COBOL files, various databases, consolidated screen definition, form definition etc. that constitute COBOL application.



This manual explains the migration of COBOL source programs, COBOL libraries, and COBOL files.

COBOL Source Programs and Libraries Migration

COBOL source programs and libraries, which are created within scope of COBOL international standards, can be migrated relatively easy. However, if you are using migration source OS or a specific function of the third-party COBOL, then you must modify it according to migration source OS and COBOL product specifications.



When source programs and libraries are described in common scope of NetCOBOL and the third-party COBOL, it is easy to migrate just by recompile and re-linkage.

However, if you use the following source programs and libraries, you cannot migrate just by recompiling and re-linkage.

- Including third-party COBOL extended specifications
- Incompatible items with NetCOBOL are described

NetCOBOL provides "pre-compiler source conversion function" to support migration.

The pre-compiler source conversion function supports source migration from third-party COBOL to NetCOBOL, by converting a source program written in the third-party COBOL syntax, into a source program which can be compiled by NetCOBOL. Using this function, you can automatically convert source programs that can be compiled using third-party COBOL compilers. Migration can be done efficiently.

This product supports the source conversion from the following COBOL supplier:

- Micro Focus COBOL

For migration of COBOL source programs and libraries, refer to "Chapter 2 Migrating COBOL Source Program and COBOL Libraries."

Migrating COBOL Files

When the COBOL files are created by applications written in the third-party COBOL syntax, NetCOBOL cannot access these files.

NetCOBOL provides "COBOL file migration tool", which converts these files into a format accessible by NetCOBOL.

This product supports the COBOL file conversion from the following COBOL supplier:

- Micro Focus COBOL

For migration of COBOL files, refer to "Chapter 3 Migration COBOL File."

Chapter 2 Migrating COBOL Source Program and COBOL Libraries

2.1 Pre-compiler Source Conversion Function

2.1.1 Overview

The pre-compiler source conversion function supports source migration from third-party COBOL to NetCOBOL by converting a source program written in the third-party COBOL syntax, into a source program which can be compiled by NetCOBOL. (Hereafter, the source written in the third-party COBOL syntax is referred to as "pre-conversion source program.")

Using this function, you can automatically convert source programs that can be compiled using third-party COBOL compilers. And migration can be done efficiently.

However, it cannot convert all source programs. Some incompatible specifications must be corrected manually. In addition, the pre-conversion source programs must not have any original compile errors.

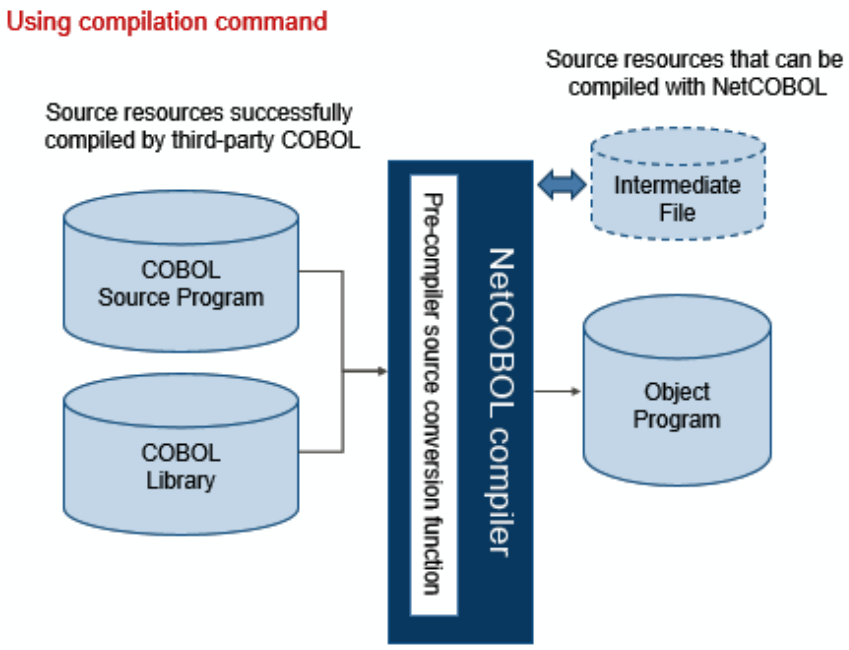
This product supports the source conversion from the following COBOL supplier:

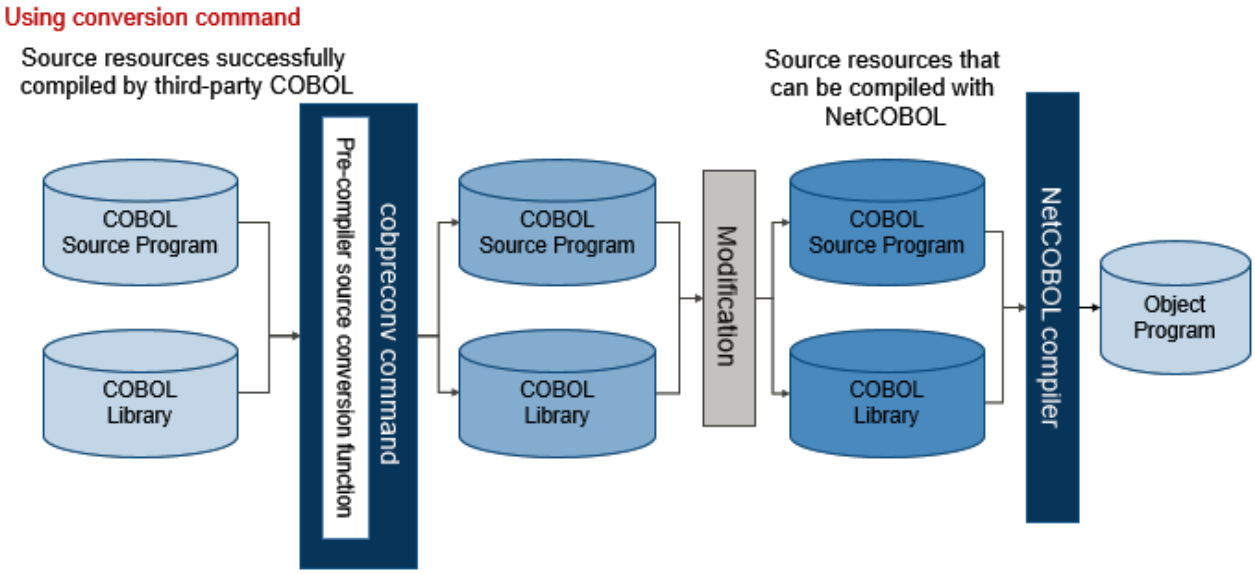
- Micro Focus COBOL

2.1.1.1 Migration Style

There are two migration styles:

- Use the NetCOBOL compiler to convert and compile the program all at once. (This is "Using compilation command.")
- Use the conversion command to convert the program. (This is "Using conversion command.")





Select a method depending on development method after migration and the target resources.

Migration style	Overview	Programs to be developed, executed and maintained	How to use
Using compilation command	Use the compilation command (cobol command). The command does the following: 1. Converts the pre-conversion source programs into source programs which can be compiled by NetCOBOL. 2. Compiles the converted source programs. Compile error message displays line number of pre-conversion source program. The builder error Go To function supports error jump in the pre-conversion source program. Also as for debugging, debug the pre-conversion source program.	Pre-conversion source program	Specify the -CV option to the cobol command. Specify the compiler option PRECONV.
Using conversion command	Use the conversion command (cobpreconv command). The command does the following: 1. Converts the pre-conversion source programs into source programs which can be compiled by NetCOBOL.	Post-conversion source program	Refer to "2.1.3.2.3 Using cobpreconv Command."

2.1.1.2 Features of Migration Methods

This section, explains the program resources and development style suited for each migration method.

Using compilation command

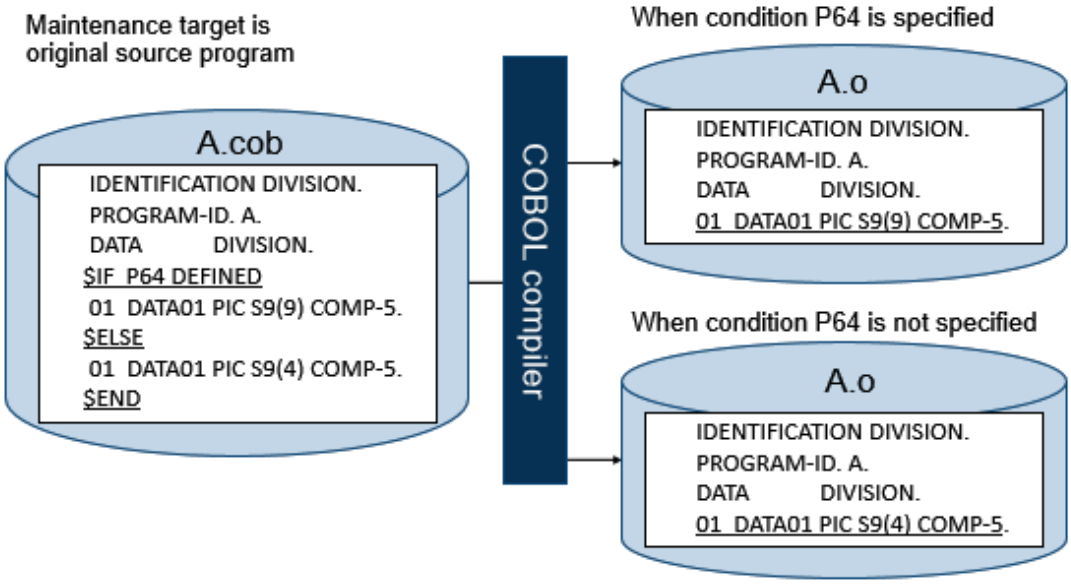
Program resources include the following:

- For resources that do not require logic modification

  If the program resources to migrate do not have such incompatibilities, migration can be done effectively by using compilation command. The compilation command converts and compiles the program all at once.

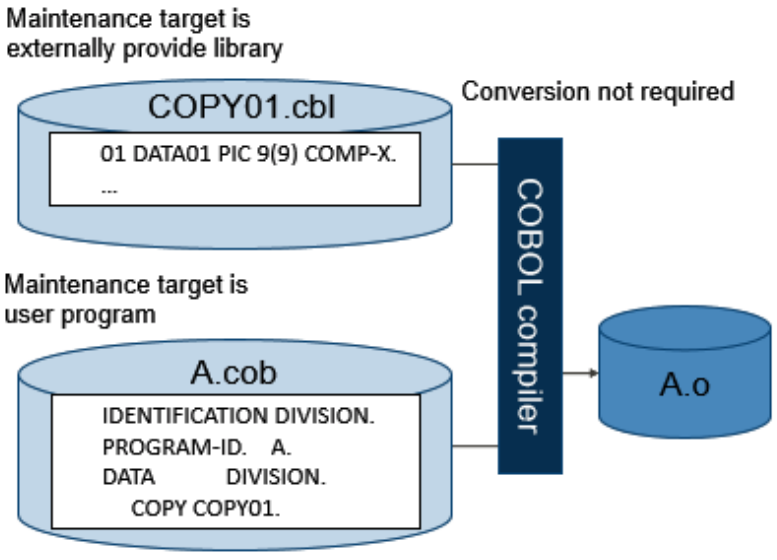
- For resources defining the conditional compilation

  You can maintain the source program before conversion. As a result, the source program can be migrated with maintainability preserved.



- For resources that contain source or libraries provided by related products such as database

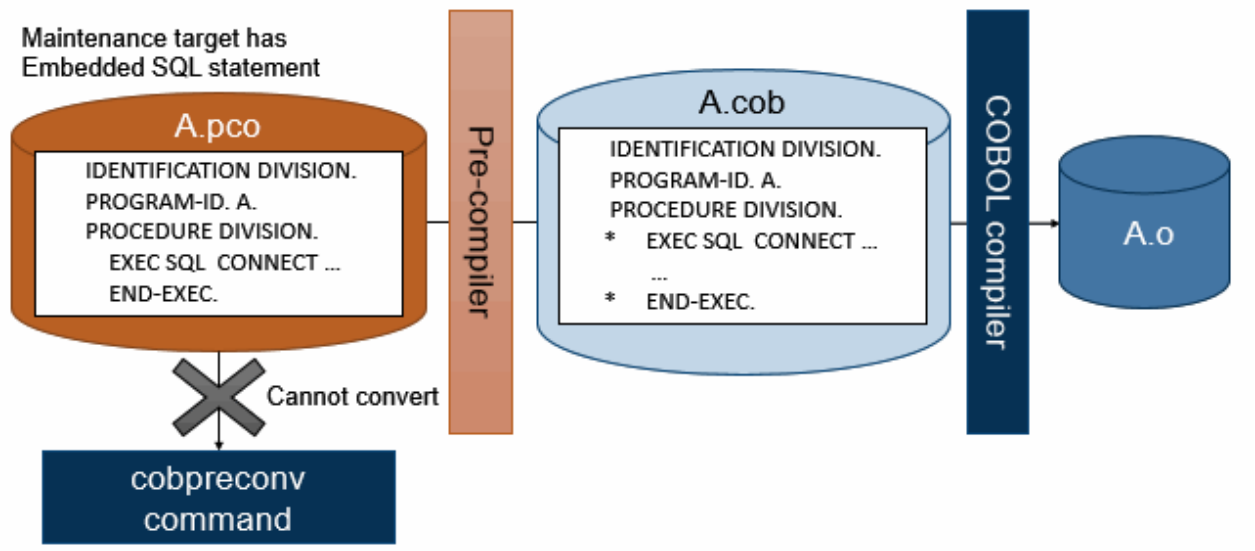
  You can compile without externally modifying the provided resources. As a result, the externally provided resource can be migrated with maintainability preserved.



- For database access program resources using pre-compiler

  The source before the pre-compiler expansion contains database specific syntax. Therefore, correct conversion may not be possible. In addition, the source after the pre-compiler expansion may contain an incompatible specification between NetCOBOL and the third-party COBOL.

It is recommended to migrate such program resources using compilation command.



Using conversion command

Program resources include the following:

- For resources that require logic modification during migration

  When the resource contains an incompatible specification, use the following steps:

  1. Use the command to convert mechanically.
  2. Check the logic on the post-conversion source program. Correct any logic errors.

- For resources that require source modification after migration

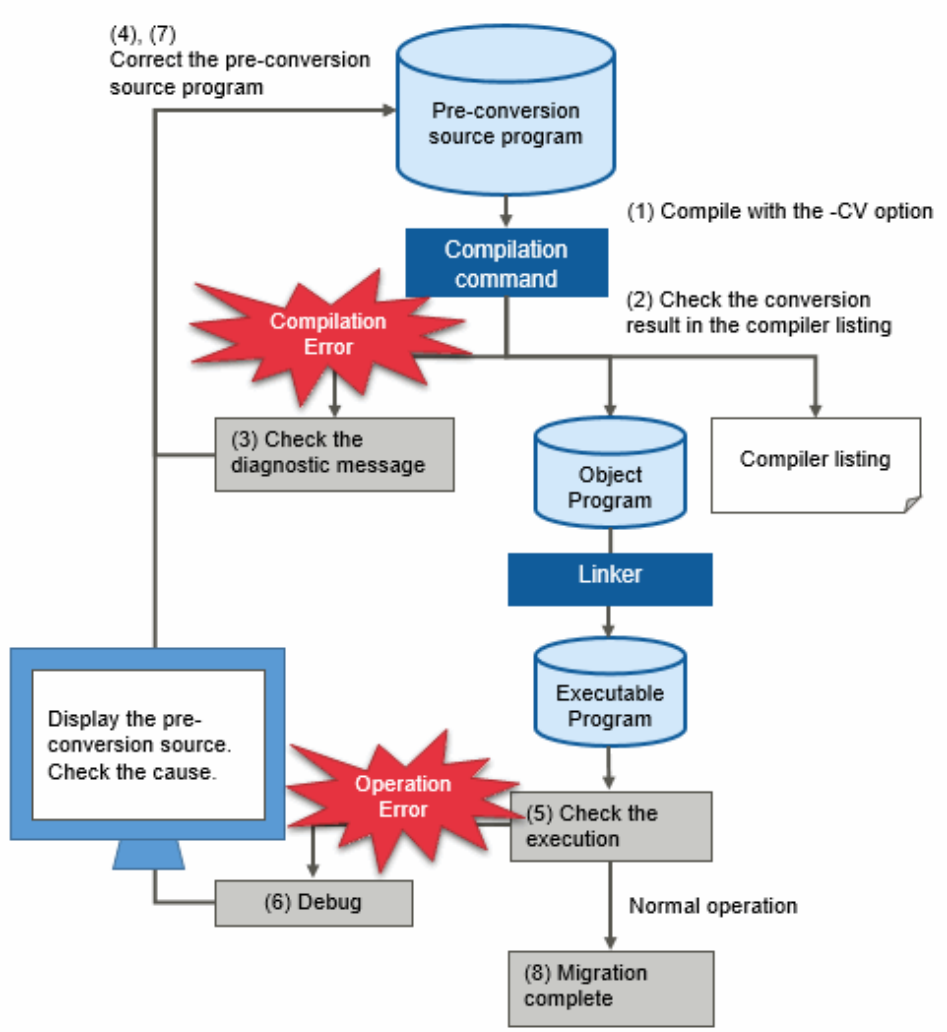
  Compile time is reduced because the source programs do not need to be converted each time they are compiled.

  In pre-compiler source conversion function, source programs mixed with third-party COBOL syntax and NetCOBOL syntax may not be converted correctly. In such a case and when major correction is required, it is recommended to migrate such program resources by using conversion command.

2.1.1.3 Migration Flow

Using compilation command

The following is the migration flow when using compilation command.



1. Compile the pre-conversion source program with -CV option.
   The pre-conversion source program is compiled by NetCOBOL compiler with the pre-compiler source conversion function enabled.
2. The conversion result is written to the compiler listing. The post-conversion source is written in the source program listing.
3. When a compile error occurs, a diagnostic message is output.

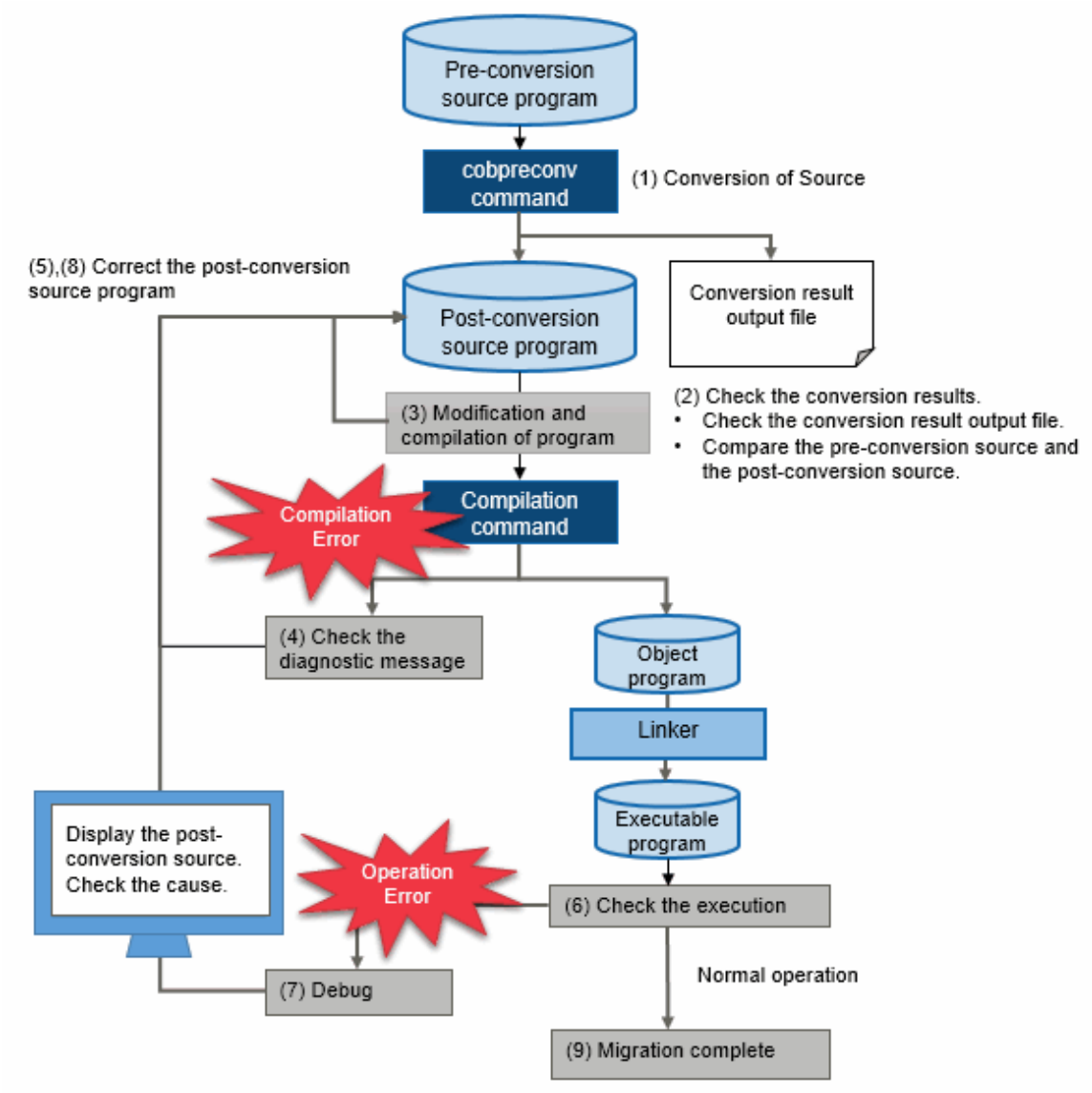
   The line number of the pre-conversion source program is displayed in the diagnostic message line number.
4. Check the diagnostic messages and correct the pre-conversion source program.

   You can use the error Go To function to correct the pre-conversion source program.
5. Check the operation of the executable program.
6. When an operation error occurs, perform debugging on the pre-conversion source program.
7. After the cause of the error is found, correct the pre-conversion source program.
8. Check that the operation of the executable program is normal. Migration is complete.

Using conversion command

The following is the migration flow when using conversion command.

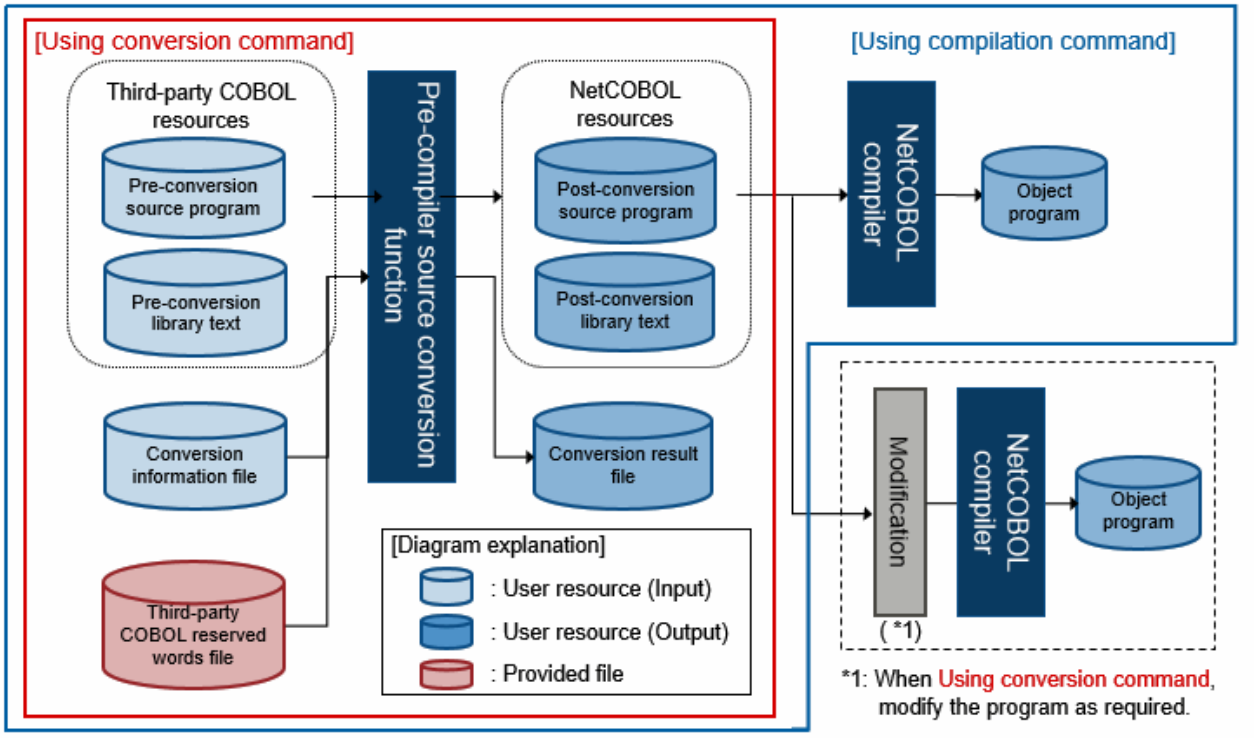
The work after converting with the conversion command is same as normal development work.



1. Convert the pre-conversion source program with cobpreconv command.
   The post-conversion source program and the conversion result file are written.
2. Check the conversion result. You can check in the following way:
   - Check the conversion result file.
   - Compare the pre-conversion source and the post-conversion source.
3. When a logic correction is required, correct the post-conversion source program.
4. When a compile error occurs, the diagnostic message is output. The line number of the post-conversion source program is displayed in the diagnostic message line number.
5. Check the diagnostic messages and correct the post-conversion source program.
   You can use the error Go To function to correct the post-conversion source program.
6. Check the operation of the executable program.
7. When an operation error occurs, perform debugging on the post-conversion source program.
8. After the cause of the error is found, correct the post-conversion source program.
9. Check that the operation of the executable program is normal. Migration is complete.

2.1.2 Input-output Files

Input-output of pre-compiler source conversion function is as follows.



Using compilation command: NetCOBOL resources are compiled and the object program is created.

Using conversion command: NetCOBOL resources are compiled at a different step.

The Input-output files used by the pre-compiler source conversion function are shown below.

Usage	Input-output file	Description	Conditions to use / create
Input	Pre-conversion source program	Third-party COBOL resource. This is converted by the pre-compiler source conversion function.	Required
	Pre-conversion library text	Third-party COBOL resource. When describing a COPY statement in a pre-conversion source program, this is converted by the pre-compiler source conversion function.	Required when converting a source program describing COPY statement
	Conversion information file	Specifies conversion rules for pre-compiler source conversion function. Depending on the content of the pre-conversion resources, user must create new file when migrating.	Required to change the conversion specification
	Third-party COBOL reserved words file (*1)	Definition file for managing the third-party COBOL reserved words.	Required
Output	Post-conversion source program	A file that converted a pre-conversion source program according to the pre-compiler source conversion function rules. NetCOBOL compilation resource.	Always written

Usage	Input-output file	Description	Conditions to use / create
		Using compilation command Creates the post-conversion source program from the pre-conversion source program using the pre-compiler source conversion function. Next, it compiles the post-conversion source program and creates an object program. Normally, use of this file is optional.	
	Post-conversion library text	File that converted a pre-conversion library text according to the pre-compiler source conversion function requirements. NetCOBOL compilation resource. Using compilation command The content of post-conversion library text is expanded to post-conversion source program. The post-conversion library text file is not written.	Using compilation command None Using conversion command Written when COPY statement is not expanded (*2)
	Conversion result file	File that contains the results from the pre-compiler source conversion function. Used to confirm the converted lines and content of the conversion from the line number and conversion ID. Using compilation command The conversion results are written to the compiler listing.	Written when output is specified (*3)

*1: The third-party COBOL Reserved words file is provided by NetCOBOL. It can be customized as required. For more details, refer to "2.1.3.5 Reserved Words File for Third-party COBOL ." Customizing the file is optional.

*2: **Using conversion command -** You can choose whether to expand the library text included by COPY statements to the post-conversion source program or not. The default does not expand and does not generate separately as post-conversion library text.

*3: To confirm the conversion portion and contents, refer to "2.1.3.4.2 Specify to Write the Conversion Result File." Default does not create a file.

2.1.3 How to Use

This section explains how to use the pre-compiler source conversion function.

2.1.3.1 Input User Resources

Prepare the following user resources as input resources.

2.1.3.1.1 Pre-conversion Source Program

Prepare a source program to migrate from third-party COBOL. For the source program, check the following information.

- COPY statement

  When COPY statement is defined, specify the library text to include. Refer to "2.1.3.1.2 Pre-conversion Library Text."

- Compiler directive of third-party COBOL

  If you are using a compiler directive, you must do one of the following:

  - Items which can be supported as conversion option

    Specify the conversion options in the conversion information file. Refer, "2.1.3.1.3 Conversion Information File."

  - Items which can be supported as a compiler option

    - **Using compilation command**

      Specify all corresponding compiler options when converting.

    - **Using conversion command**

      Only compiler options which are valid at conversion must be specified when converting.

      Even if you specify a compiler option which is not valid, an error does not occur.

    For details on how to specify the compiler option, refer to "2.1.3.4.3 Compiler Option ."

  - Items to which NetCOBOL does not have compatible options

    You must investigate and consider whether the same operation can be performed as before migration.

For details on compatibility with each compiler directive, refer to "A.3 Compiler Directive of Third-party COBOL."

2.1.3.1.2 Pre-conversion Library Text

When a COPY statement is described in pre-conversion source programs, specify the library text to include.

The following is different for third-party COBOL and NetCOBOL:

- Search order of library file directory
- Default value of extension

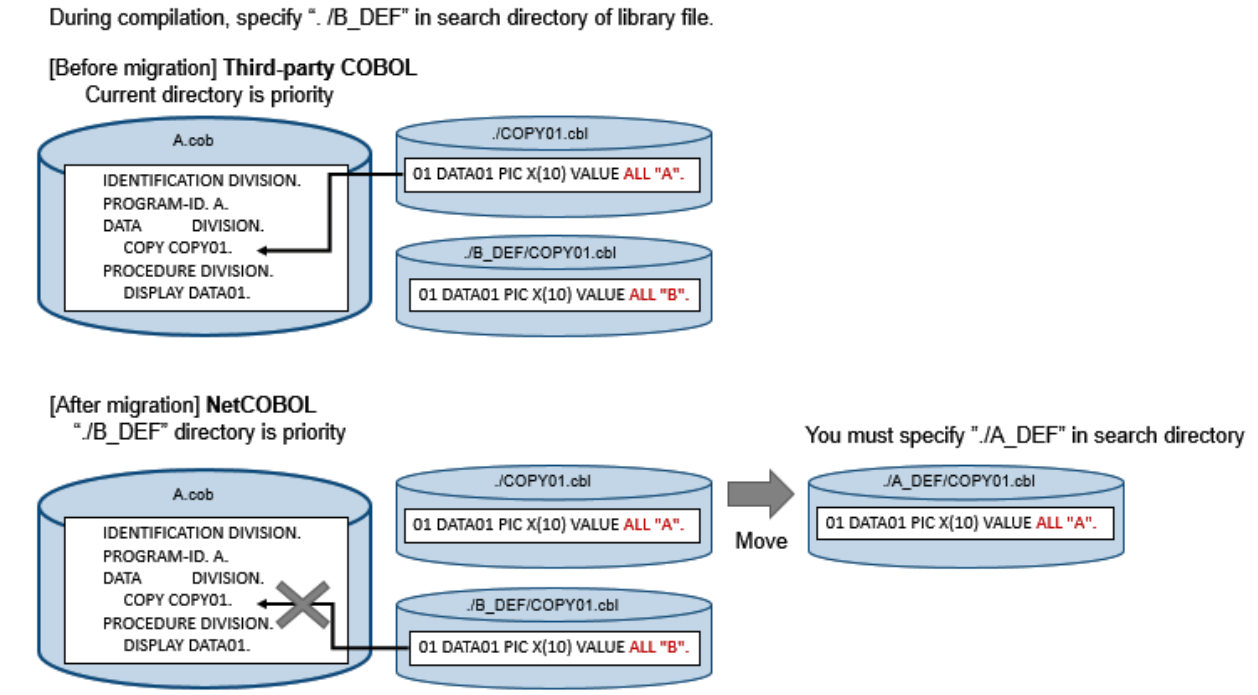
Especially in the following cases, library files different from those before migration may be included.

- Files with the same name exist in multiple directories
- Varying library files with different extensions exist

Review the specification method of the library files and program resources directory structure as per the requirements, so that correct library files can be included.

 Example

Example: When a library file of same name exists in current directory and subdirectory



For details on how to specify in NetCOBOL, refer to "2.1.3.4.1 Specify the Library Files Directory and Extension."

2.1.3.1.3 Conversion Information File

In the pre-compiler source conversion function, use conversion options to specify the conversion requirements.

The conversion information file is a text file which specifies the conversion option.

A conversion information file is required when converting the following pre-conversion source program.

- Compiler directives are used with compatible compiler options
- $IF statement (*1) is described, which compares the constant value
- $IF statement (*1) is described, which specifies compiler directive

*1: For details on $IF statement, refer to conversion ID m0101.

Format of Conversion Information File

This section explains the format of conversion information file.

W Character code for conversion information file is ANSI code page (Shift JIS) or UTF-8 with BOM.

L Character code for conversion information file is created with the same character code as the pre-conversion source programs.

Each line of the conversion information file is delimited with a line feed character. The line feed characters correspond to that of Windows (0x0d0a) and UNIX (0x0a).

A conversion information file is comprised of the common section and several sections.

Common section

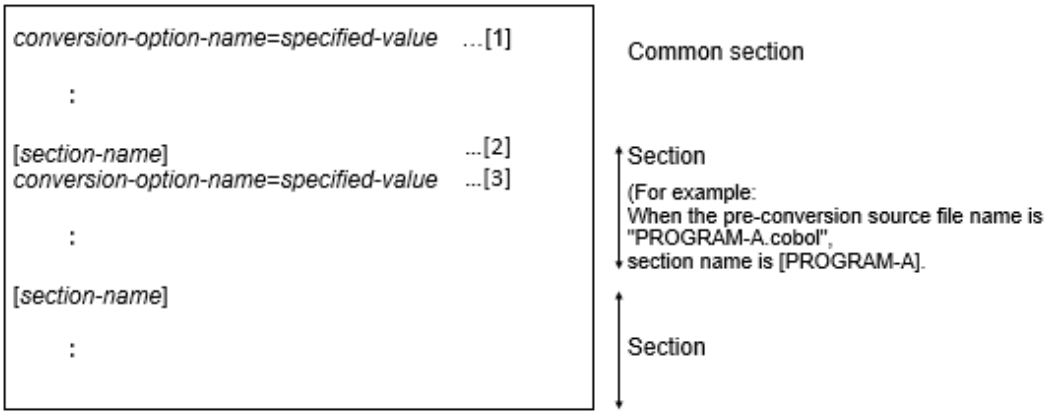
Describe the conversion specifications common to each program.

Section

Describe the conversion specification for each program.

Section name is described by enclosing with square brackets ("[]"). The content up to the next section name is regarded as one section.

Content of conversion information file is shown in the given below.



"Conversion-option-name = specified-value", is called as an "option information."

[1] Describe the conversion option information common to each program.

The conversion option information written in this section is effective for all programs.

If the same conversion option information is specified in the common section and another section having the same name as the name of the program to be converted, the contents in the named section overrides the common section.

[2] Indicates the beginning of conversion option information for each program. The section name is the pre-conversion source file name without an extension. Note that the section name is not a PROGRAM-ID.

[3] Describe the conversion option information for each program.

See

For conversion option details, refer to "A.4 Specification Information of Conversion Information File."

Note

- Only one option specification is allowed per line.
- Do not specify more than one conversion option with the same name in one section. If more than one conversion option with the same name is specified, the operation is not guaranteed.

Comment of Conversion Information File

When the line starts with a semicolon (;), the section from the semicolon up to the line feed character is recognized as a comment. Excessive commented lines may cause decreased performance due to skipping of these lines.

2.1.3.2 Execute Conversion Processing

When you have prepared the user resources to enter, then execute conversion processing for pre-compiler source conversion function. Conversion processing is done in one of the following ways.

- Using cobol command as compilation command
- Using NetCOBOL Studio as compilation command
- Using cobpreconv command as conversion command

2.1.3.2.1 Using cobol Command

Note

For [Winx64], COBOL command runs on NetCOBOL command prompt.

Specify -CV option when using pre-compiler source conversion function. Specify -CV option in the following format.

```
-CV{conversion-type} [,conversion-information-file-name]
```

- conversion-type

  Specify type of syntax in which pre-conversion source program is described. The specified value is below.

Conversion Type	Meaning
m	Micro Focus COBOL syntax

- conversion-information-file-name

  To change the conversion specifications, specify the conversion information file path name after a comma.

For details, refer to "NetCOBOL User's Guide."

Example

Example 1: Example of specification when converting and compiling one source file

W

```
C:\COBOL>COBOL -M -CVm -I.\cpy test.cob
```

Command action

1. Convert test.cob and create a post-conversion source program.
2. Compile the post-conversion source program as main program. The object file (test.obj) is output.

L

```
$ cobol -M -CVm -I./cpy test.cob
```

Command action

1. Convert test.cob and create a post-conversion source program.
2. Compile the post-conversion source program as main program. The object file (test.o) and executable file (a.out) are output.

Example 2: Example of a message output when a conversion error occurs

```
$ cobol -M -CVm -I.\cpy test.cob
test.cob 8: JMN0503I-S  PRCV-ER108S COBOL library 'COPY001' could not be found.
STATISTICS: HIGHEST SEVERITY CODE=S, CONVERTED PROGRAM UNIT=1.
JMN0020I-U The source program file is empty. Compilation terminated.
```

The cobol command outputs information such as the conversion result, compilation results and diagnostic messages. These are output to the following output destinations:

- For [Winx64]: Command prompt
- For [Linux64]: Standard error output

W

2.1.3.2.2 Using NetCOBOL Studio

1. Create a new COBOL project.

2. Register pre-conversion source program as a COBOL source file in the project.
3. Specify compiler option PRECONV from the **Compiler options** page.

   When you want to change conversion specifications, specify path name of conversion information file in **PRECONV compile options** dialog.

   For details on setting options, refer to "Setting compile options" in "NetCOBOL Studio User's Guide."
4. Build the project.

   The diagnostic message is displayed in the **Problems** view. If you do either of the following operations, the pre-conversion source program is displayed in the editor.

   - Double-click the compile error information in the **Problems** view
   - On the compile error information in the **Problems** view, select **Go To** from the context menu

See

Refer to the following:

- "PRECONV(Determines whether to use the pre-compiler source conversion function)" in "NetCOBOL User's Guide"
- "Building a COBOL program" in "NetCOBOL Studio User's Guide"
- "Correcting Compilation Errors" in "NetCOBOL Studio User's Guide"

2.1.3.2.3 Using cobpreconv Command

Note

For [Winx64], the cobpreconv command runs from NetCOBOL command prompt.

Converts pre-conversion source program into a source program which can be compiled by NetCOBOL.

Specify cobpreconv command, in the following format.

```
cobpreconv -CV{conversion-type}[,conversion-information-file-name] -o output-directory-
name [-f] [compilation-command-option] file-name ...
```

- -CV{*conversion-type*}[,*conversion-information-file-name*]

  The specification format of -CV option specification is same as COBOL command.
- -o *output-directory-name*

  Specify the output directory for the post-conversion source program. The specified directory must exist.
- -f

  When -f option is specified, if file with the same name as the post-conversion source program and post-conversion library text exists, the file is overwritten.
- *compilation-command-option*

  W Only -I, -P, -dp, and -WC can be specified.

  L Only -I, -P, -dp, -i, and -WC can be specified.
- *file-name*

  Specify the pre-conversion source file. Multiple files can be specified.

For more details, refer to "cobpreconv Command" in "NetCOBOL User's Guide."

Example

Example1: Convert one source file

W

```
C:\COBOL>COBPRECONV -CVm -I.\cpy -o.\out test.cob
```

L

```
$ cobpreconv -CVm -I./cpy -o./out test.cob
```

Command action

1. Convert test.cob.
2. Writes the post-conversion source program as test.cob in the directory specified by -o.

Example2: Convert multiple source files

W

```
C:\COBOL>COBPRECONV -CVm -I.\cpy -o.\out test1.cob test2.cob
```

L

```
$ cobpreconv -CVm -I./cpy -o./out test1.cob test2.cob
```

Command action

1. Convert test1.cob and test2.cob.
2. Write the post-conversion source program as test1.cob and test2.cob in the directory specified by -o.

The cobpreconv command outputs information such as the conversion result, compilation results and diagnostic messages. These are output to the following output destinations:

- For [Winx64]: Command prompt
- For [Linux64]: Standard error output

2.1.3.3 Confirmation of Conversion Result

This section explains the conversion result confirmation methods.

2.1.3.3.1 Severity code

Pre-compiler source conversion function displays a severity code for each conversion item.

Severity code	Conversion processing	Meaning
I	Convert	Converts in accordance with conversion specifications.
W	Convert	Converts in accordance with conversion specifications. However the user must confirm that the conversion results are as intended.
E	No conversion	There is no conversion. The conversion item must be corrected manually.

When an error occurs in conversion processing, the following message is output based on the severity.

Severity Code	Level
I	Informational Message
W	Warning error
E	Recoverable error
S	Serious error
U	Unrecoverable error

The highest severity code is determined from the converted item severity and the detected error severity.

2.1.3.3.2 Conversion ID

In pre-compiler source conversion function, a conversion ID is added for each of the conversion items.

Conversion ID is written to the conversion result file together with the pre-conversion source program or library text line number.

Conversion ID consists of the following five alphanumeric characters.

```
nxxyy
```

- n: Specifies conversion type specified with -CV option
- xx: Specifies the following function-specific ID

  01: General rules

  02: Identification division

  03: Environment division

  04: Data division

  05: Procedure division

  06: Source text manipulation

  yy: Serial number

2.1.3.3.3 Conversion Message

In either of the following cases, conversion messages are output.

a. Conversion items with severity code W or higher was detected

b. An error was detected during the conversion process

Conversion messages are output to the following output destinations:

- For [Winx64]: Standard output
- For [Linux64]: Standard error output

Format of conversion message

```
Message-number Message-text
```

- Message-number

  ```
  PRCV-xxxxxS
  ```

  - PRCV: Specifies a pre-compiler source conversion function message
  - xxxxx :
    - In case of a, it is set to "Conversion ID." For Conversion ID, refer to "2.1.3.3.2 Conversion ID."
    - In case of b, it is set to "ERnnn." nnn is message serial number.
  - S: Specifies severity code

    For the severity code, refer to "2.1.3.3.1 Severity code ."

Output Format

Using conversion command

Depicts conversion message in following format.

Message format

```
file-name  line-number : conversion-message
```

Using compilation command

Conversion messages are embedded as message text of compiler message for each severity code.

Severity Code	Compiler Message
I	JMN0500I-I
W	JMN0501I-W
E	JMN0502I-E
S	JMN0503I-S
U	JMN0504I-U

The file name and line number are displayed as given below.

```
file-name  line-number : JMN05xxI-S conversion-message
```

2.1.3.3.4 Post-conversion Source Program

When the highest severity code is less than E, conversion is complete. The post-conversion source program is written. The output file name is the same as the pre-conversion source program in the following directory.

Using compilation command

Write output to the .preconv directory under pre-conversion source program directory.

When there is no .preconv directory, a new directory is created. You can change the default output directory.

For details on how to change, refer to "WORKDIR" of "Conversion information file."

Using conversion command

Write to the directory specified by -o option. For details, refer to "cobpreconv Command" in "NetCOBOL User's Guide."

Note

- Using conversion command

  When post-conversion source program already exists, the default is to display an error. When overwriting the post-conversion source program, specify -f option.

- Using compilation command

  - Every time you compile, the post-conversion conversion source program is overwritten.
  - When highest severity code is higher than S, then the following error is output.

    ```
    JMN0020I-U The source program file is empty. Compilation terminated.
    ```

2.1.3.3.5 Post-conversion Library Text

Using compilation command

When a COPY statement is described in the pre-conversion source program, content of post-conversion library text is expanded to the post-conversion source program. Post-conversion library text is not created.

Using conversion command

When a COPY statement is described in the pre-conversion source program, by default, post-conversion library text is created to the directory specified by -o option. By specifying the conversion option, you can expand the contents of library

source text to converted source program. For details on how to change, refer to "EXPAND-COPY"of "A.4 Specification Information of Conversion Information File."

Note

- When a path name is specified in a COPY statement, the post-conversion library text may be output to a directory different from directory specified by -o option. For details, refer to "2.1.4.1.3 Handling of Library Text File ."
- When post-conversion library text already exists, the default is to display an error. When overwriting the post-conversion library text, specify -f option.
- When copying one library text from multiple pre-conversion source programs, for the same reason as above, an error is output in the conversion of the second and subsequent programs. When specifying -f option, the post-conversion library text is overwritten. However, the content identity is not checked.

  When you want to create library texts with varying conversion specifications, change the output directory or specify contents of library text to expand to post-conversion source program.

2.1.3.3.6 Conversion Result File

To check the conversion details and conversion contents, create the conversion result file.

For details on how to generate the file, refer to "2.1.3.4.2 Specify to Write the Conversion Result File."

The following is written to the conversion result file.

- Pre-conversion source program line number
- Conversion ID

Check the conversion content from conversion ID referring to "A.2 Conversion Item Details."

For details on the conversion result file, refer to "Appendix B Conversion Result File."

See

Using compilation command

When you compile using compiler listing output (-P option or compiler option PRINT) and compiler option SOURCE, source program listing is written to the conversion result file. At this time, the post-conversion source program is written to source program listing.

2.1.3.4 Specify Compiler Option and Environment Variable

For specification of source program interpretation during of conversion, the pre-compiler source conversion function supports the same format as the COBOL compiler option and environment variables.

2.1.3.4.1 Specify the Library Files Directory and Extension

When converting a source program describing a COPY statement, specify the following as required.

Confirm the search order of the directory and file extension since these may differ from the third-party COBOL.

Specify the library files directory

Library file search order is as follows.

1. -I option
2. Compiler option LIB (Using compilation command)
3. Environment variable COB_COBCOPY
4. Current folder

1. -I option
2. Environment variable COBCOPY
3. Current directory

See

For details, refer to "NetCOBOL User's Guide."

For [Winx64]:

- "-I(Specify the library file folder)"
- "LIB(Library file folder specification)"
- "Setting Environment Variables"

For [Linux64]:

- " -I (Specification of library file directory)"
- "COBCOPY (Specification of library file directory)"

Specify library files extension

When changing extension of library files, specify the environment variable COB_LIBSUFFIX.

You can specify multiple extensions in environment variables. When multiple extensions are used, extensions are searched in the order specified. Also, when "None" is specified files without extensions are searched.

```
COB_LIBSUFFIX
```

When environment variable is not specified, the search order is as follows.

1. Extension .CBL
2. Extension .COB
3. Extension .COBOL

For details of how to specify environment variable, refer to "Setting Environment Variables" in "NetCOBOL User's Guide."

1. Extension .cbl
2. Extension .cob
3. Extension .cobol

For details of how to specify environment variable, refer to "COB_LIBSUFFIX (Specification of extension of library file)" in "NetCOBOL User's Guide."

Specify library text search condition

You can specify upper and lower case handling in environment variable COB_COPYNAME when searching the library file name.

The following can be specified when describing the COPY statement in the source program.

- text-name
- text-name literal
- library-name
- library-name literal

By specifying one of the following values in the environment variable, all upper-case or all lower-case library file names can be searched.

- Upper

  Searches for all upper-case file names.

- Lower

  Searches for all lower-case file names.

- Default

  Searches for file name of "*library-text-name*.*lower-case-extension*" format.

  The text-name and library-name is non case-sensitive. However, text-name literal and library-name literal is case-sensitive.

2.1.3.4.2 Specify to Write the Conversion Result File

The pre-compiler source conversion function writes the conversion result to a file. This file is called "Conversion result file."

Using compilation command

NetCOBOL compiler writes a compiler listing to check the compilation result.

When you specify the output of compiler listing, the conversion result is written to the compiler listing file.

For [Winx64]

Compiler option	Meaning	Reference NetCOBOL User's Guide
- -P option - Compiler option PRINT (When using compilation command)	To specify output and output path for various compiler listing	- "-P(Compile listing file name)" - "PRINT(whether compiler listing should be output and the output destination specification)"
-dp option	To specify folder for compiler listing file	"-dp(Specify the compile list file folder)"

For [Linux64]

Compiler option	Meaning	Reference NetCOBOL User's Guide
-P option	To specify the file name of the compiler listing	"-P (Specification of the file name of the compiler listing)"
-dp option	To specify directory for compiler listing file	"-dp (Specification of compiler listing file directory)"

Note

Conversion result file is the same file as the compiler listing.

When checking the specification method with reference to "NetCOBOL User's Guide", read "compiler listing file" with "conversion result file" replaced.

2.1.3.4.3 Compiler Option

Compiler option used during conversion. Compiler option without description does not impact conversion results.

Compiler option	Meaning	Usage
ENCODE RCS	Runtime code system	Used to determine the runtime code system.
RSV	Type of reserved words	To replace NetCOBOL reserved words.
SCS	Code system of the source file	Used to determine the pre-conversion source program and library text code system.
SRF	Reference format type	Used to determine the pre-conversion source program reference format (*1).
TAB	Tab handling	Used to determine the tab code width described in pre-conversion source program. According to this specification, the tab code is converted to 8 or 4 column spaces. This specification is valid even for tab values in the literal.
CURRENCY	Currency symbol	Used to determine the character for currency symbol in the pre-conversion source program.

*1: The second operand for the compiler option SRF cannot be specified. COBOL source program and library file reference format are considered to be the same.

The priority sequence and specification method of compiler option, is as given below.

W

1. Using compiler directing statement (@OPTIONS) in source program
2. Using -WC option
3. Using environment variable COB_OPTIONS
4. Using command option
5. Using **Compiler Option** page of NetCOBOL Studio (When using compilation command)

For details on how to specify the compiler option, refer to "Compiler Option Specification Formats" in "NetCOBOL User's Guide."

L

1. Using compiler directing statement (@OPTIONS) in source program
2. Using -WC option
3. Using compiler options specified by the -WC option of the environment variable COBOLOPTS
4. Using command option
5. Using environment variable COBOLOPTS
6. Using compiler options in the options file specified by the -i option

For details on how to specify the compiler option, refer to "Compiler Option Specification Formats" in "NetCOBOL User's Guide."

2.1.3.5 Reserved Words File for Third-party COBOL

The pre-compiler source conversion function determines whether a word in the pre-conversion source program is a reserved word or not according to the definition in the reserved words file.

By default, entire words listed as reserved words in the third-party COBOL are considered as reserved words.

In addition, in the third-party COBOL, reserved words types can be changed by compiler directives.

When there is a word you do not want to treat as a reserved word (for example, a user-defined word is included in the reserved words list), you must customize the reserved words file to remove that word.

In the pre-compiler source conversion function, the following customization methods are supported.

- For individual reserved words, to specify words excluded from reserved words

  Edit the reserved words file directly. For details, refer to "Appendix C How to Customize Reserved Words."

2.1.3.6 Automatically Enabled Compiler Option

The pre-compiler source conversion function generates the compiler option control information in the post-conversion source program so that compiler options compatible with the default behavior of third-party COBOL are automatically enabled at compile time.

The compiler option control information (#OPTIONS) is an information to notify NetCOBOL compiler of the compiler options. Compiler options specified by #OPTIONS has a lower priority than other methods of specifying compiler options.

For automatically enabled compiler options, refer to "2.1.4.1.2 Compiler Options Suitable for Migration."

2.1.4 Conversion Specification

This section explains the specifications for each conversion type.

Conversion type shows syntax that describes the pre-conversion source program using the -CV option.

The pre-compiler source conversion function supports the following conversion type.

Conversion type	-CV specified-value
Micro Focus COBOL	m

2.1.4.1 Micro Focus COBOL

2.1.4.1.1 Conversion Specification

Refer to "Appendix A Migration Reference (from Micro Focus COBOL)."

2.1.4.1.2 Compiler Options Suitable for Migration

NetCOBOL supports the compiler options suitable for third-party COBOL migration programs. Specify it as required.

Compiler option	Meaning	#OPTIONS
ARITHMETIC(31)	Uses 31 digits operation calculation mode.	ineffective (*1)
BINARY(BYTE)	Assigns the elementary item of binary data to an area length of 1 to 8 bytes.	effective (*2)
FLAG(E)	Displays diagnostic messages of only E-level or higher.	ineffective (*3)
FILELIT(ENV)	Handle file-identifier literal as a file-identifier.	ineffective
INITVALUE(20)	Initialize items without a VALUE phrase in WORKING-STORAGE section data with alphanumeric blanks (X "20").	effective
MF(ALL)	Specify Micro Focus compatible mode.	effective

Compiler option	Meaning	#OPTIONS
SRF(FIX)	Specify the reference format of a COBOL source program and library with fixed-length.	effective

effective: Automatically effective compiler options

ineffective: Automatically ineffective compiler options

Notes:

- *1: Required when using a numeric item which exceeds 18 digits. However, ARITHMETIC (31) and BINARY (BYTE) or BINARY (WORD, MLBOFF) cannot be specified at the same time. Confirm impact on area length of binary item and specify ARITHMETIC (31) and BINARY (WORD, MLBON) at the same time.
- *2: When compiler option BINARY (WORD) is specified at conversion, it is ineffective. The value of specified compiler option BINARY is enabled.
- *3: It is not automatically enabled as sometimes diagnostic messages related to incompatibility during migration are output. Further, for FLAG, there is a second operand for excluding the diagnostic messages which are interpreted the same as third-party COBOL. It is recommend to specify FLAG (I, MF) which can only check NetCOBOL specific diagnostic messages.

2.1.4.1.3 Handling of Library Text File

In NetCOBOL and third-party COBOL, specification method and search order are different. When the library text directory and file extension are correctly stated, the same pre-migration library text file can be used.For details, refer to "2.1.3.4.1 Specify the Library Files Directory and Extension."

However, a part of COPY statement format is incompatible.

In pre-compiler source conversion function, these incompatibilities are handled so that the same library files can be entered.

For [Winx64]

Format of COPY statement				Path name of pre-conversion library text (*1)	When COPY statement is not expanded	
					File name of post-conversion library text (*2)	Notes on compilation (*4)
Text-name			COPY f1.	dir2\F1.cp2	F1.cp2	-
Text-name literal	No path name	Without extension	COPY "f1".	dir2\f1.cp2	f1.cp2	(*4a)
		With extension	COPY "f1.cp1"	dir2\f1.cp1	f1.cp1	-
			COPY f1.cp1.	dir2\F1.CP1	F1.CP1	(*4b)
	Relative path	Without extension	COPY "dir1\f1".	dir1\f1.cp2	f1.cp2	(*4a)
			COPY dir1\f1.	DIR1\F1.cp2	F1.cp2	(*4a)

Format of COPY statement				Path name of pre-conversion library text (*1)	When COPY statement is not expanded	
					File name of post-conversion library text (*2)	Notes on compilation (*4)
		With extension	COPY "dir1\f1.cp1".	dir1\f1.cp1	f1.cp1	-
			COPY dir1\f1.cp1.	DIR1\F1.CP1	F1.CP1	(*4b)
	Absolute path	Without extension	COPY "D:\dir1\f1".	D:\dir1\f1.cp2	f1.cp2	(*4c)
			COPY D:\dir1\f1.	D:\DIR1\F1.cp2	F1.cp2	(*4b)
		With extension	COPY "D:\dir1\f1.cp1".	D:\dir1\f1.cp1	f1.cp1	(*4c)
			COPY D:\dir1\f1.cp1.	D:\DIR1\F1.CP1	F1.CP1	(*4b)
With IN/OF			COPY f1 IN subdir. COPY f1 OF subdir. COPY f1 IN "subdir". COPY f1 OF "subdir".	subdir\F1.cp2(*3)	F1.cp2	(*4d)

For [Linux64]

Format of COPY statement				Path name of pre-conversion library text (*1)(*5)	When COPY statement is not expanded	
					File name of post-conversion library text (*2)(*5)	Notes on compilation (*4)
Text-name			COPY f1.	dir2/F1.cp2	F1.cp2	-
Text-name literal	No path name	Without extension	COPY "f1".	dir2/f1.cp2	f1.cp2	(*4a)
		With extension	COPY "f1.cp1"	dir2/f1.cp1	f1.cp1	-
			COPY f1.cp1.	dir2/F1.CP1	F1.CP1	(*4b)
	Relative path	Without extension	COPY "dir1/f1".	dir1/f1.cp2	f1.cp2	(*4a)

Format of COPY statement				Path name of pre-conversion library text (*1)(*5)	When COPY statement is not expanded	
					File name of post-conversion library text (*2)(*5)	Notes on compilation (*4)
			COPY dir1/f1.	DIR1/F1.cp2	F1.cp2	(*4a)
		With extension	COPY "dir1/f1.cp1".	dir1/f1.cp1	f1.cp1	-
			COPY dir1/f1.cp1.	DIR1/F1.CP1	F1.CP1	(*4b)
	Absolute path	Without extension	COPY "/work/dir1/f1".	/work/dir1/f1.cp2	f1.cp2	(*4c)
			COPY /work/dir1/f1.	/WORK/DIR1/F1.cp2	F1.cp2	(*4b)
		With extension	COPY " /work/dir1/ f1.cp1".	/work/dir1/f1.cp1	f1.cp1	(*4c)
			COPY /work/dir1/f1.cp1.	/WORK/DIR1/F1.CP1	F1.CP1	(*4b)
With IN/OF			COPY f1 IN subdir. COPY f1 OF subdir.	SUBDIR/F1.cp2 (*3)	F1.cp2	(*4d)
			COPY f1 IN "subdir". COPY f1 OF "subdir".	subdir/F1.cp2 (*3)	F1.cp2	

Common notes for [Winx64] and [Linux64]

- *1: Result when specifying "dir2" as path name and "cp2" as extension for library files.
- *2: Create a file with same name as pre-conversion library text inside directory specified by -o option.
- When copying multiple library texts with the same name that exist in different paths, by default, an error occurs when the second or subsequent post-conversion library texts are written. When specify -f option, the post-conversion library text is overwritten. However, the content identity is not checked. In this case, specify contents of library text to expand to post-conversion source program.
- *3: When specifying a COPY statement containing a path name, if specified library file does not exist, directory specified in path name (in this case "dir2") of library file would be searched.
- *4: If the COPY statement is not expanded in post-conversion source program, then post-conversion source program may need to be modified before compilation.
  - 4a: When library name without extension is specified in text name literal, add an extension.
  - 4b: When the quotation marks is omitted in text name literal, add a quotation marks.
  - 4c: When it contains an absolute path name, change the path name.
  - 4d: When IN/OF is used in COPY statement, delete IN/OF to use the text name literal with the path name.

- *5: This is the result when you do not specify environment variable COB_COPYNAME. To specify handling upper and lower case, use the environment variable COB_COPYNAME. For details, refer to "2.1.3.4.1 Specify the Library Files Directory and Extension."

2.1.5 Notes

2.1.5.1 NetCOBOL Studio Notes (when using compilation command)

2.1.5.1.1 Editor

Default Reference Format

In NetCOBOL Studio, the default reference format of the editor is a variable length format.

To use other reference formats, refer to "Reference Formats" in "NetCOBOL Studio User's Guide."

Maximum Length of Source Program

Maximum line length for NetCOBOL variable length format is 251 bytes. Characters described in a range exceeding this number are treated as comments. Source code described in editor which exceeds 251 bytes is displayed as comment.

When you migrate a source which exceeds 251 bytes in NetCOBOL, use the "continuation line", and edit source program, so that it does not exceed 251 bytes.

For details of "continuation line", refer to "Continuation of Lines" in "NetCOBOL Language Reference."

Highlighting Reserved Words

In reserved word highlighting, the editor cannot highlight words that are only a reserved word in the third-party COBOL syntax. Reserved words for third-party COBOL are recognized and can be highlighted only if they are also reserved words in NetCOBOL.

Creating Conversion Information File

NetCOBOL Studio cannot create UTF-8 conversion information file with BOM.

To create UTF-8 conversion information file with BOM, use tools other than NetCOBOL Studio.

For details of conversion information file character code, refer to "2.1.3.1.3 Conversion Information File."

2.1.5.1.2 Debugging Function

Add Data Item in Watch View

- When using pre-compiler source conversion function, the data name in pre-conversion source program may be renamed. Using "Add Data Item" using data name in source program may fail. For details on the format of the data name to be renamed, refer to following conversion item.
  - m0107 (Convert to NetCOBOL non-reserved words. (Specify "-RV" option))
- When using pre-compiler source conversion function, level-number for data item of pre-conversion source program might be changed. In this case, subscripts may be required for data items to be added to Watch view. For details on the format of the level number of data item is changed, refer to following conversion item.
  - m0403 (Convert level-number 01 or 77 data items with OCCURS clause specified)

Add Conditional Expression in Watch View

When using pre-compiler source conversion function, conditional expressions in the pre-conversion source program might be changed. Using "Add Condition" for the conditional expression in source program, may not display the conditional expression. For details on format of conditional expressions is changed, refer to following conversion item.

Furthermore, only conditional expressions that NetCOBOL supports can be used.

- m0509 (Convert relational conditions - comparing numeric data item and alphanumeric data item)
- m0510 (Convert relational conditions - comparing numeric data item and nonnumeric literal)
- m0511 (Convert relational conditions - comparing national data item and nonnumeric literal)
- m0512 (Convert relational conditions - comparing national data item and alphanumeric data item)

2.1.5.2 Pre-compiler Source Conversion Function

In pre-compiler source conversion function, conversion is based on descriptions in "A.1 Conversion Item List" out of incompatible specifications of NetCOBOL and third-party COBOL. For other incompatible specifications, no conversion occurs. At this time, no conversion message is output. Check the diagnostic messages output during compilation and correct them manually.

2.1.6 Limitation

The limitations related to pre-compiler source conversion function are mentioned below.

Cord System

- Character code Unicode

Compiler option ENCODE(UTF8), RCS(UTF16), and SCS(UTF8) cannot be specified.

If the locale is Unicode, specify compiler options SCS(SJIS) and ENCODE (SJIS).

Reference Format

- Free format

  Compiler option SRF(FREE) cannot be specified.

- Change of reference format

  You cannot change the reference format inside the file.

  Also, the library file cannot have a different reference format than the COBOL source program.

How to Code a COBOL Source Program and Library

- Source program where each division header is omitted

  When a source program that does not have division header or section header is used as the pre-conversion source file, conversion may not be performed correctly.

- Conversions related to IDENTIFICATION DIVISION comment entries

  In the following cases, the character strings in comment entry may be changed according to the conversion rules for user defined words.

  - When a word that is a NetCOBOL reserved word and is not a third-party COBOL reserved word is described.

- Converting NetCOBOL reserved words

  When a word that is a NetCOBOL reserved word and a third-party COBOL context-sensitive word, is used as a user defined word, the word is not converted.

- Conversion of source files with several compilation units

  In pre-compiler source conversion function, when you specify a source program containing several compilation units, it will not be converted correctly.

- DECIMAL-POINT IS COMMA clause

  The pre-compiler source conversion function does not support DECIMAL-POINT IS COMMA clause. Commas as a decimal point may be replaced with blanks.

- When conversion items are on same line as COPY statement, the conversion item may not be converted correctly
- Debugging lines

  In pre-compiler source conversion function, the debugging lines are treated as comment lines. If there is a description to be converted in the debugging lines, conversion is not performed.
- When the PICTURE character-string is continued to a continuation line, the conversion result may not be correct
- Concatenation expression

  Concatenation expression may not be converted correctly.
- Abbreviating a combined relation condition

  When the arithmetic expression is described in the abbreviated combined relation condition, the conversion item may not be converted correctly.
- RENAMES entry

  The RENAMES entry is not subject to conversion to comparison or moving.

Compiler Option

- Compiler option FLAG

  For the conversion messages output by the pre-compiler source conversion function, you cannot specify the diagnostic message level to display.
- Compiler option ALPHAL

  In the pre-compiler source conversion function, regardless of how the compiler option ALPHAL is specified, it is used as follows:

  - User defined words

    The lower case letters are differentiated from upper case letters. If the word is not used in a consistent manner in the program, it will not be converted correctly.
  - Reserved words

    Lower case letters and upper case letters are treated as equal.
- Compiler option LINECOUNT

  For the conversion result file written by the pre-compiler source conversion function, the page is not changed even if LINECOUNT is specified.
- Compiler option LINESIZE

  The pre-compiler source conversion function writes the conversion result file as one continuous line whether or not LINESIZE is used.
- Compiler option NUMBER (when using compilation command)

  In pre-compiler source conversion function, you cannot specify the compiler option NUMBER. Enable the compiler option NONUMBER and compile.

Other

- Environment variables COBMSG_FORMAT

  For the conversion messages output by the pre-compiler source conversion function, the format of diagnostic message cannot be changed.
- Compilation with multiple source file specified (when using compilation command)

  When the pre-compiler source conversion function is enabled, you cannot specify multiple source file in the compilation command. To convert multiple files, execute the compilation command more than once.

Chapter 3 Migration COBOL File

When the COBOL files are created by applications written in the third-party COBOL syntax, NetCOBOL cannot access these files.

NetCOBOL provides "COBOL file migration tool" which converts these files into a format which is accessible by NetCOBOL.

This product supports the COBOL file conversion from the following COBOL supplier:

- Micro Focus COBOL

  This tool is called "Micro Focus COBOL File Migration Tool."

3.1 Micro Focus COBOL File Migration Tool

3.1.1 Function Overview

Micro Focus COBOL File Migration Tool converts the COBOL file created by an application written in Micro Focus COBOL syntax into a format accessible by NetCOBOL.

This tool operates on Windows.

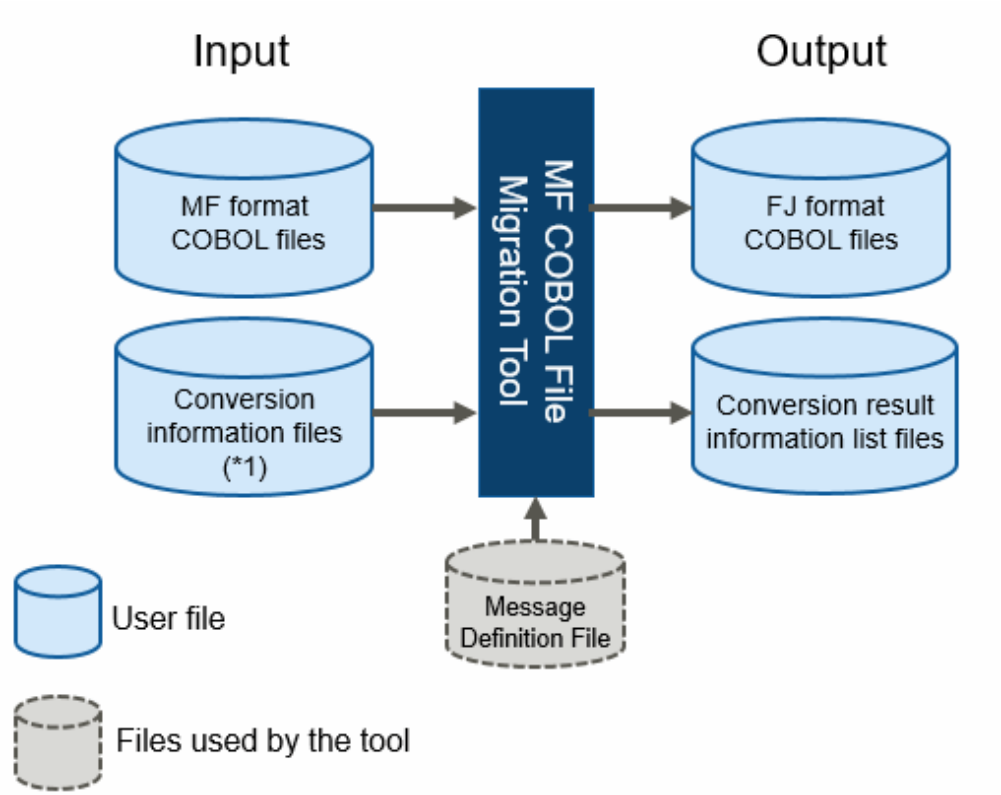
It is called as follows:

- Micro Focus COBOL File Migration Tool is called "MF COBOL File Migration Tool "
- COBOL file format created by an application written in Micro Focus COBOL syntax is called "MF format"
- COBOL file format that NetCOBOL can access is called "FJ format"

This tool converts COBOL files in MF format to COBOL files in FJ format.

However, in regards to index files, conversion is done by taking the output of this tool as an intermediate file and further using "COBOL File Utility" to convert to FJ format index files.

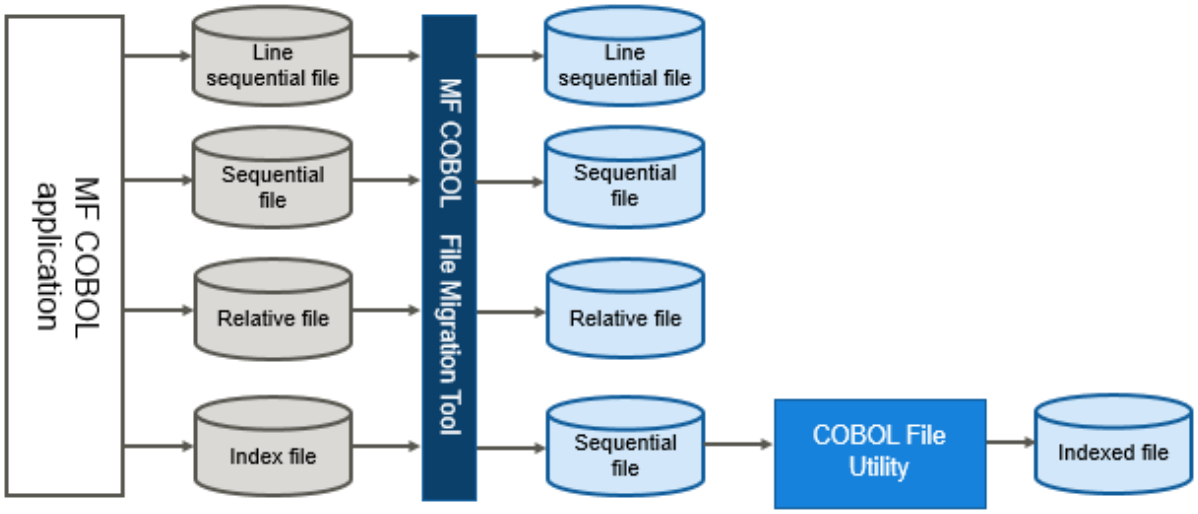
The input-output of this tool is given below.



*1: This conversion information file is different from the "conversion information file" used by pre-compiler source conversion function. In this chapter, the conversion information file refers to input-output file of the file migration tool.

3.1.2 Positioning of Tool

Positioning of this tool is shown below.



3.1.3 Input-Output File

The following are the input-output files used by this tool.

	Resource name	Storage directory	File name	Usage	Contents	File format
1	Conversion information file	Optional	Optional	Input (Required)	File information, conversion options, etc.	Text file
2	MF format COBOL file	Optional	Optional	Input (Required)	Pre-conversion file	Line sequential file Sequential file Relative file Indexed file
3	FJ format COBOL file	Optional	Same name as pre-conversion file	Output	Post-conversion file However, indexed file is converted to variable length sequential file	Line sequential file Sequential file Relative file
4.	Conversion result information list	Optional	*pre-conversion filename*.LST	Output	Conversion result information.	Text file

Description of Each Files

1. Conversion information file

   Specify the information required to convert a file. For details, refer to "3.1.4.3 Conversion Information File."

2. MF format COBOL file

   Specify the MF format COBOL file. For details, refer to "3.1.5 Conversion Specifications."

3. FJ format COBOL file

   FJ format COBOL file is created by this tool. For details, refer to "3.1.4 How to Use."

4. Conversion result information list

   The conversion result information is written by this tool.

   When there is an extension in pre-conversion file name, the extension is changed to ".LST". If the file name does not have an extension, ".LST" is added. For details, refer to "3.1.5 Conversion Specifications."

3.1.4 How to Use

This section describes how to use MF COBOL File Migration Tool.

3.1.4.1 To Start the Tool

1. Click **Start**, select *Fujitsu NetCOBOL product name* > **MFTOFJFC**. **MF COBOL File Migration Tool** appears.
2. Specify pre-conversion file in **Pre-conversion File Name** edit box.
   When specifying multiple files, click **Refer,** and the pre-conversion file specification dialog appears. You can select multiple files.
3. Specify conversion information file in **Conversion information File** edit box. Specification of conversion information file is required.
4. Click **Convert** or **OK**. If you click **Cancel**, then this tool will exit without converting.

Note

When the pre-conversion file name and conversion information file name are input from keyboard, you must enter the full path.

For details of operation, refer to help on start-up screen.

L

3.1.4.2 Command Format

```
mftofjfb pre-conversion-file-name conversion-information-file
```

- pre-conversion-file-name

  Specify the path name of the pre-conversion file.

- conversion-information-file

  Specify the path name of the conversion information file.

3.1.4.3 Conversion Information File

You must create a conversion information file.

W

Contents of conversion information file

```
* Comment line                                                    [1] (Optional)
OUT-FILE-DIR=Post-conversion-file-folder-name\                    [2] (Required)
OUT-LIST-DIR=Conversion-result-information-list-file-folder-name\ [3] (Optional)
ORGANIZATION={LINSEQ | SEQ | REL | IDX}                           [4] (Required)
RECORD-FORMAT={F | V}                                             [5] (Required)
RECORD-LENGTH=Maximum-record-length                               [6] (Required)
ZONED-DECIMAL=Position,Number-of-digits,{ LEADING | TRAILING }    [7] (Optional)
INDEX-DATA-ITEM=Position                                          [8] (Optional)
```

Note: Indicate that one of the values given in the braces can be selected.

Rules

- Conversion information file must be in text format.
- Character code of file is ANSI code page (Shift JIS).
- One line must be 256 bytes or less.
- When the first character is "*", the line is considered to be a comment line.
- Specification of each option can be in any order.
- For [2] and [3], the folder names must be specified by full path ("drive:\~"). If it is on network, then it must be specified by UNC path ("\\computer-name\shared-folder\~").
- The folder name must end with "\".
- Option [7] and [8] can be specified more than once. Other options can be specified only once. If same option is specified more than once, contents of option specified later would become effective.

L

Contents of conversion information file

```
* Comment line                                                    [1] (Optional)
OUT-FILE-DIR=Post-conversion-file-directory-name\                 [2] (Required)
OUT-LIST-DIR=Conversion-result-information-list-file-directory-name\ [3] (Optional)
ORGANIZATION={LINSEQ | SEQ | REL | IDX}                           [4] (Required)
RECORD-FORMAT={F | V}                                             [5] (Required)
RECORD-LENGTH=Maximum-record-length                               [6] (Required)
```

```
ZONED-DECIMAL=Position,Number-of-digits,{ LEADING | TRAILING }        [7] (Optional)
INDEX-DATA-ITEM=Position                                              [8] (Optional)
```

Note: Indicate that one of the values given in the braces can be selected.

Rules

- Character code of file is Unicode (UTF-8).
- One line must be 256 bytes or less.
- When the first character is "*", the line is considered to be a comment line.
- Specification of each option can be in any order.
- For [2] and [3], the directory names must be specified by full path ("drive:\~"). If it is on network, then it must be specified by UNC path ("\\computer-name\shared-directory\~").
- The directory name must end with "\".
- Option [7] and [8] can be specified more than once. Other options can be specified only once. If same option is specified more than once, contents of option specified later would become effective.

Description of Options

[1] Comment line

The line where first character is"*", is a comment line. The content is free.

[2] OUT-FILE-DIR=*Post-conversion-file-directory-name*\

Specify a directory name for saving the post-conversion file. The post-conversion file is written in the same directory with file name same as pre-conversion file.

- If a file with same name already exists in storage directory, the file is overwritten.
- You cannot specify same directory as pre-conversion file. If specified, an error occurs.

[3] OUT-LIST-DIR=*Conversion-result-information-list-file-directory-name*\

Specify a directory name for saving the conversion result information list file.

When the pre-conversion file name has an extension, the extension is changed to ".LST". If the file name does not have an extension, ".LST" is added.

If this option is omitted, then conversion result information list file is written in the directory specified by "OUT-FILE-DIR=".

If a file with same name already exists in storage directory, the file is overwritten.

[4] ORGANIZATION={LINSEQ | SEQ | REL | IDX}

Specify file organization.

- LINSEQ: Line sequential file
- SEQ: Sequential file
- REL: Relative file
- IDX: Indexed file

[5] RECORD-FORMAT={F | V}

Specify record format.

- F: Fixed length
- V: Variable length

[6] RECORD-LENGTH=*Maximum-record-length*

Specify record length. For fixed length, specify its length. For variable length, specify maximum length.

[7] ZONED-DECIMAL=*Position*,*Number-of-digits*,{ LEADING | TRAILING }

Specify information on signed zoned decimal data item without the SEPARATE phrase.

This information is required to convert sign (internal format). When there are multiple items, specify this option for each item. When specifying multiple items, there are no restrictions in the order.

- Position: Specify relative position in a record (taking the start position as 1).
- Number of digits: Specify the total number of digits in the signed zoned decimal data item.
- Specify the position of sign.
  - LEADING : There is sign on the leftmost side
  - TRAILING: There is sign on the rightmost side

[8] INDEX-DATA-ITEM=*Position*

Specify relative position within record of index data item (taking start position as 1). When there are multiple items, specify this option for each item. When specifying multiple items, there are no restrictions in the order.

3.1.4.4 Conversion of Index Files

When the pre-conversion COBOL file is an index file, this tool converts it to an intermediate file (variable length sequential file). You must then convert this intermediate file to FJ format Index file using "COBOL File Utility."

For details "COBOL File Utility", refer "NetCOBOL User's Guide."

3.1.4.5 Error Messages

When the conversion process cannot be continued, an error message is output.

In such a case, check contents of error message, take appropriate measures, and then perform the conversion again.

The error messages are output in following cases:

- When there is a mistake in description of contents of conversion information file
- When there is an access error in input-output file
- When the limit of this tool is exceeded
- When there is contradiction between content of pre-conversion file and content specified by conversion option

If this tool terminates abnormally due to an irrecoverable application error, check each input file again.

3.1.4.6 File Conversion Examples

Here are some examples for converting files:

Example 1: To convert the sequential file (Fixed length)

1. Prepare a file to convert.

   In this example, convert the following file.

   [Input file name]

W

```
C:\MFDATA\SEQFIL01.FIL
```

L

```
/home/mfdata/SEQFIL01.FIL
```

[Output directory name]

The post-conversion file and the conversion result information list are written.

W

```
C:\FJDATA
```

L

```
/home/fjdata/
```

[File specification]

- Sequential file (fixed length)
- Maximum record length: 80 bytes
- Zoned decimal item: 2 items
- Index data item: 1 item
- Record format

```
01  SEQ-FILE.                                              Relative-position
 02  DATA-1    PIC 9(4).                                                 1
 02  DATA-2    PIC S9(4).                  *1: Signed zoned decimal      5
 02  DATA-3    PIC X(4).                                                 9
 02  DATA-4    USAGE IS INDEX.             *2: Index data item          13
 02  DATA-5    PIC X(4).                                                17
 02  DATA-6    PIC S9(8) SIGN IS LEADING.  *3: Signed zoned decimal      21
 02  DATA-7    PIC X(52).                                               29
```

2. Create the conversion information file.

W

File name

```
 C:\MFDATA\SEQFIL01.INF
```

It has the following contents.

```
*** Sequential file (Fixed length) conversion ***
OUT-FILE-DIR=C:\FJDATA\
ORGANIZATION=SEQ
RECORD-FORMAT=F
RECORD-LENGTH=80
ZONED-DECIMAL=5,4,TRAILING          (*1)
ZONED-DECIMAL=21,8, LEADING         (*3)
INDEX-DATA-ITEM=13                  (*2)
```

L

File name

```
/home/mfdata/SEQFIL01.INF
```

It has the following contents.

```
*** Sequential file (Fixed length) conversion ***
OUT-FILE-DIR=/home/fjdata/
ORGANIZATION=SEQ
RECORD-FORMAT=F
RECORD-LENGTH=80
```

```
ZONED-DECIMAL=5,4,TRAILING          (*1)
ZONED-DECIMAL=21,8, LEADING         (*3)
INDEX-DATA-ITEM=13                  (*2)
```

3. Execute the conversion tool

   Execute this tool. For how to start, refer to "3.1.4.1 To Start the Tool."

4. Check the conversion results

   Following files are created. Please check the conversion result information list.

W
- C:\FJDATA\SEQFIL01.LST (Conversion result information list)
- C:\FJDATA\SEQFIL01.FIL (FJ format sequential file (Fixed length))

L
- /home/fjdata/SEQFIL01.LST (Conversion result information list)
- /home/fjdata/SEQFIL01.FIL (FJ format sequential file (Fixed length))

Example 2: To convert the indexed file (Variable length)

1. Prepare a file to convert.

   In this example, convert the following file.

   [Input file name]

W
```
C:\MFDATA\IDXFIL01.FIL
```

L
```
/home/mfdata/IDXFIL01.FIL
```

   [Output directory name]

W
```
C:\FJDATA              (Post-conversion file)
C:\FJDATA\LIST         (Conversion result information list)
```

L
```
/home/fjdata/          (Post-conversion file)
/home/fjdata/LIST      (Conversion result information list)
```

   [File specification]

   - Indexed file (Variable length)
   - Maximum record length: 80 bytes
   - Zoned decimal item: 2 items
   - Index data item: None
   - Record format:

     In the MF format, it is considered that items specified with OCCURS DEPENDING ON are acquired in large size.

```
01  IDX-FILE.                                                Relative position
  02  DATA-1    PIC 9(4).                                          1
  02  DATA-2    PIC S9(4).             *1: Signed zoned decimal    5
  02  DATA-3    PIC X(4).                                          9
  02  DATA-4    PIC X(4).                                         13
  02  DATA-5    PIC X(4).                                         17
  02  DATA-6    PIC S9(8) SIGN IS LEADING.  *2: Signed zoned decimal  21
  02  DATA-7    OCCURS 1 TO 10 TIMES DEPENDING ON DATA-7-DEP.     29
   03 DATA-7-ITEM   PIC X(10).
  02  DATA-8    PIC S9(4).                 *3: Signed zoned decimal  129
```

2. Create the conversion information file.

W

File name

```
C:\MFDATA\IDXFIL01.INF
```

It has the following contents.

```
*** Indexed file (Variable length) conversion ***
OUT-FILE-DIR=C:\FJDATA\
OUT-LIST-DIR=C:\FJDATA\LIST\
ORGANIZATION=IDX
RECORD-FORMAT=V
RECORD-LENGTH=80
ZONED-DECIMAL=5,4,TRAILING         (*1)
ZONED-DECIMAL=21,8, LEADING        (*2)
ZONED-DECIMAL=129,4, TRAILING      (*3)
```

L

File name

```
/home/mfdata/IDXFIL01.INF
```

It has the following contents.

```
*** Indexed file (Variable length) conversion ***
OUT-FILE-DIR=/home/fjdata/
OUT-LIST-DIR=/home/fjdata/LIST/
ORGANIZATION=IDX
RECORD-FORMAT=V
RECORD-LENGTH=80
ZONED-DECIMAL=5,4,TRAILING         (*1)
ZONED-DECIMAL=21,8, LEADING        (*2)
ZONED-DECIMAL=129,4, TRAILING      (*3)
```

3. Execute the conversion tool

   Execute this tool. For how to start, refer to "3.1.4.1 To Start the Tool."

4. Check the conversion results

   Following files are created.

W

- C:\FJDATA\LST\IDXFIL01.LST (Conversion result information list)
- C:\FJDATA\IDXFIL01.FIL (FJ format sequential file (variable length))

L

- /home/fjdata/LIST/IDXFIL01.LST (Conversion result information list)
- /home/fjdata/IDXFIL01.FIL (FJ format sequential file (variable length))

5. Convert a sequential file (variable length) to index file

W

Using COBOL File Utility

1. Click **Start**, select *Fujitsu NetCOBOL product name* > **COBOL File Utility**.
2. The **COBOL FILE UTILITY** appears. On the **Commands** menu, click **Load**.
3. The **LOAD** dialog appears. Input the file information.
   - **Input** text box: Specify a sequential file name (variable length).
   - **Output** text box: Specify an indexed file name that is created by this utility.

- Output file information:

  Organization: Click **IXD**. (As indexed file)

  Record format: Click **VAR** or **FIX**. In the **Record length** text box, specify maximum record length (Also in the case of fixed length format).

4. Click **KEY**.
5. The **KEY INFORMATION** dialog appears. Input the index key information.
6. Click **OK**.

   Indexed files are created.

Using COBOL File Utility Command

Use the load command (cobfload) to convert a sequential file (variable length) into index file.

3.1.5 Conversion Specifications

This section explains about conversion specification for the MF COBOL File Migration Tool.

3.1.5.1 Conversion Content

Conversion from MF format file to FJ format file, can be broadly divided into following two types.

- Conversion of data format
- Conversion of file format

Conversion of Data Format

The following shows the incompatibilities between MF data format and FJ data format.

Data format/attribute	Incompatible content	MF	FJ
Signed zoned decimal (Without TRAILING SEPARATE / LEADING SEPARATE)	The Internal representation of sign portion is different.	Value of sign portion Positive number: X'3x' Negative number: X'7x'	Value of sign portion Positive number: X'4x' Negative number: X'5x'
Index data item	Attributes are different.	COMP format	COMP-5 format

When the above data items are included in the file record, conversion to FJ format is required. Specify the conversion information in the conversion information file. If not specified, conversion is not performed.

Conversion of File Format

The following shows the compatibilities between MF file format and FJ file format.

File Type	Record Format	Compatibility	Conversion	Comments
Line sequential file	-	Yes	Not required	*1, *2
Sequential file	Fixed length	Yes	Not required	*1, *3
	Variable length	No	Required	
Relative file	Fixed length	No	Required	
	Variable length	No	Required	

File Type	Record Format	Compatibility	Conversion	Comments
Indexed file	Fixed length	No	Required	When this tool uses an indexed file, it writes a sequential file (variable length). You must convert from this sequential file to an indexed file using COBOL File Utility. For details, refer to "3.1.4 How to Use."
	Variable length	No	Required	

*1: The signed zoned decimal item without TRAILING SEPARATE and LEADING SEPARATE must be converted.

*2: For the line sequential file (line sequential files before LEVEL II COBOL version 2.1 or earlier) that requires to specify "-N" as runtime switch of Micro Focus, this must be converted as well as the variable length sequential file.

*3: If there are indexed data items, they must be converted.

For converting files, please specify the file information in the "conversion information file." Refer to "3.1.4.3 Conversion Information File."

3.1.5.2 Conversion Result Information List

The tool writes the information of the converted file in the "conversion result information list." If an error occurs during conversion, the error information is also written in "conversion result information list."

The conversion result information is shown as below.

Example output of conversion result information list

W

```
MF COBOL FILE MIGRATION TOOL   V12.2.0    DATE 01/05/2020  TIME 11:22:33
< ERROR INFORMATION >
  error messages                                                [1]


< FILE NAME >
  PRE-CONVERSION FILE                = C:\MFDATA\SEQFIL01.FIL       [2]
  POST-CONVERSION FILE               = C:\FJDATA\SEQFIL01.FIL       [3]
  CONVERSION INFORMATION FILE        = C:\MFDATA\SEQFIL01.INF       [4]
  CONVERSION RESULT INFORMATION LIST = C:\FJDATA\SEQFIL01.LST       [5]

< FILE INFORMATION >
  FILE TYPE = SEQUENTIAL FILE                                   [6]
  RECORD FORMAT = FIXED LENGTH                                  [7]
  MAXIMUM RECORD LENGTH = 100                                   [8]
  MINIMUM RECORD LENGTH = 30                                    [9]

< DATA CONVERSION >
  CONVERSION OF SINGED ZONED DECIMAL ITEM : EXECUTED           [10]
  CONVERSION OF INDEX DATA ITEM           : NOT EXECUTED       [11]

< INPUT-OUTPUT COUNT >
  INPUT RECORD  = 100000                                       [12]
  OUTPUT RECORD = 100000                                       [13]

CONVERSION START   : 11H 22M 33S                               [14]
CONVERSION END     : 11H 25M 00S                               [15]
CPU TIME           : 00H 02M 27S                               [16]

HIGHEST SEVERITY CODE : 00                                     [17]
```

L

```
MF COBOL FILE MIGRATION TOOL   V12.2.0    DATE 01/05/2020  TIME 11:22:33
< ERROR INFORMATION >
```

```
  error messages                                                  [1]


< FILE NAME >
  PRE-CONVERSION FILE               = /home/MFDATA/SEQFIL01.FIL    [2]
  POST-CONVERSION FILE              = /home/FJDATA/SEQFIL01.FIL    [3]
  CONVERSION INFORMATION FILE       = /home/MFDATA/SEQFIL01.INF    [4]
  CONVERSION RESULT INFORMATION LIST = /home/FJDATA/SEQFIL01.LST   [5]

< FILE INFORMATION >
  FILE TYPE = SEQUENTIAL FILE                                     [6]
  RECORD FORMAT = FIXED LENGTH                                    [7]
  MAXIMUM RECORD LENGTH = 100                                     [8]
  MINIMUM RECORD LENGTH = 30                                      [9]

< DATA CONVERSION >
  CONVERSION OF SINGED ZONED DECIMAL ITEM : EXECUTED             [10]
  CONVERSION OF INDEX DATA ITEM           : NOT EXECUTED         [11]

< INPUT-OUTPUT COUNT >
  INPUT RECORD  = 100000                                          [12]
  OUTPUT RECORD = 100000                                          [13]

CONVERSION START   : 11H 22M 33S                                  [14]
CONVERSION END     : 11H 25M 00S                                  [15]
CPU TIME           : 00H 02M 27S                                  [16]

HIGHEST SEVERITY CODE : 00                                        [17]
```

[1] Error message

If an error occurs during conversion, the error information is written. In this case, the information [3], [6] to [16] is not written.

[2] PRE-CONVERSION FILE

Indicates the pre-conversion file name.

[3] POST-CONVERSION FILE

Indicates the post-conversion file name.

[4] CONVERSION INFORMATION FILE

Indicates the conversion information file name.

[5] CONVERSION RESULT INFORMATION LIST

Indicates the conversion result information list name.

[6] FILE TYPE

Indicates the file type:

- LINE SEQUENTIAL FILE
- SEQUENTIAL FILE
- RELATIVE FILE
- INDEXED FILE

[7] RECORD FORMAT

Indicates the record format:

- FIXED LENGTH
- VARIABLE LENGTH

When the file type ([6]) is SEQUENTIAL FILE, RELATIVE FILE, or INDEXED FILE, this information is written.

[8] MAXIMUM RECORD LENGTH

Indicates the maximum record length.

[9] MINIMUM RECORD LENGTH

Indicates the minimum record length.

When the record format ([7]) is VARIABLE LENGTH, this information is written.

[10] CONVERSION OF SINGED ZONED DECIMAL ITEM

Indicates whether the signed zoned data item conversion was performed or not.

[11] CONVERSION OF INDEX DATA ITEM

Indicates whether the index data item conversion was performed or not.

[12] INPUT RECORD

Indicates the number of records entered by the tool.

[13] OUTPUT RECORD

Indicates the number of records written by the tool.

[14] CONVERSION START

Indicates the time when the conversion started.

[15] CONVERSION END

Indicates the time when the conversion ended.

[16] CPU TIME

Indicates the time required for conversion.

[17] HIGHEST SEVERITY CODE

Indicates the return value.

- 00: Completed successfully
- 04: Completed successfully (No impact on conversion)
- 12: Terminated abnormally (Conversion information error/inconsistency etc.)
- 16: Terminated abnormally (Fatal error such as file I/O error)

3.1.6 Restrictions

Following restrictions are included in this tool.

a. Maximum record length

In COBOL, maximum record length of a file is 32760 bytes. If this value is exceeded, it cannot be converted.

b. Indexed file

This tool only supports conversion of Micro Focus COBOL indexed files in a format that separates the data part and the index part.

Indexed files in other formats cannot be converted.

c. Data item with OCCURS DEPENDING ON clause (hereafter referred to as ODO clause)

When the item (*1) to be converted is in variable position, it cannot be converted.

Therefore, files which are created by specifying a compiler directive ODOSLIDE in MF COBOL cannot be converted. Please create files where item to be converted is at a fixed position and convert.

*1: The items to be converted are as follows:

- Signed zoned decimal item without SEPARATE

    - Index data item

For MF and FJ, the allocation position of data item immediately after the data item specifying ODO clause is different as follows:

- MF
    - When compiler directive NOODOSLIDE is specified or default

      It is positioned immediately after the maximum size of the ODO table.

    - When compiler directive ODOSLIDE is specified

      It is positioned immediately after current size of ODO table.
- FJ

  It is positioned immediately after current size of ODO table.

Appendix A Migration Reference (from Micro Focus COBOL)

When migrating COBOL source files and library files from Micro Focus COBOL to NetCOBOL, the following specifications are explained.

- Conversion specifications
  - Conversion item list
  - Conversion item details
- Micro Focus COBOL compiler directives
- Conversion information file requirements

A.1 Conversion Item List

The conversion ID is assigned for each conversion item.

The conversion ID is written to the conversion result file, along with pre-conversion source program or library text line number.

Table A.1 General Rules

Function	Conversion ID	Severity code	Meaning
Conditional compilation	m0101	I	Determine the condition of the $statement (IF, ELSE, END) and convert it to comment line
	m0102	I	Output text data as conversion message when $DISPLAY statement is specified
Specification of compiler option	m0105	W	Convert the $SET statement to comment line
Reserved words	m0107	I	Convert to non-reserved word for NetCOBOL. (Specify "-RV" option)
Reference format	m0113	I	Convert the line to a comment line when the first column is "*", X"00" to X"1F"
Maximum line length of source	m0118	E	Convert lines longer than 251 bytes

Table A.2 Environment Division

Function	Conversion ID	severity code	Meaning
SPECIAL-NAMES paragraph	m0305	I	Convert SWITCH 0 to 7
	m0306	W	Convert SWITCH 8
	m0307	I	Convert NUMERIC SIGN IS TRAILING SEPARATE clause
FILE-CONTROL paragraph	m0310	I	Delete EXTERNAL or DYNAMIC phrase of ASSIGN clause
	m0311	I	Create ASSIGN clause data name area for dynamic file allocation
	m0321	I	Convert split key phrase (indexed organization) of RECORD KEY clause

Table A.3 Data Division

Function	Conversion ID	severity code	Meaning
Report section	m0402	I,E	Convert named literal in report section to SYMBOLIC CONSTANT clause
OCCURS clause	m0403	I,E	Convert level-number 01 or 77 data items with OCCURS clause specified
USAGE clause	m0404	I,W	Convert numeric items from COMP-X to BINARY
	m0405	I,W,E	Convert alphanumeric data items from COMP-X to BINARY
VALUE clause	m0406	I	Convert from figurative constant ZERO or QUOTE to national nonnumeric literal (When figurative constant ZERO or QUOTE is specified for national data item)
	m0407	I	Convert from nonnumeric literal to national nonnumeric literal (When nonnumeric literal is specified for national data item)

Table A.4 Procedure Division

Function	Conversion ID	Severity code	Meaning
Comparison rules	m0509	I,E	Convert relation condition (Comparing numeric data item (binary item or packed decimal item) and alphanumeric data item)
	m0510	I,E	Convert relation condition (Comparing numeric data item (binary item or packed decimal item) and nonnumeric literal)
	m0511	I,E	Convert relation condition (Comparing national data item and nonnumeric literal)
	m0512	I,E	Convert relation condition (Comparing national data item and alphanumeric data item (or numeric edited data item or zoned decimal item))
	m0513	I	Convert relation condition (Comparing national data item and figurative constant ZERO or QUOTE)
	m0546	E	Unable to convert relation condition (In abbreviating combined relation conditions, when it is require to rewrite the left operand)
Move rules	m0514	I	Convert MOVE statement (moving from alphanumeric data item or numeric edited data item to national data item)
	m0515	I	Convert MOVE statement (moving from zoned decimal data item to national data item)
	m0516	I	Convert MOVE statement (moving from nonnumeric literal to national data item)
	m0517	I	Convert MOVE statement (moving from hexadecimal nonnumeric literal to national data item)
	m0518	I	Convert MOVE statement (moving from national data items to alphanumeric data item or numeric edited data item)
	m0519	I	Convert MOVE statement (moving from figurative constant ZERO or QUOTE to national data items)

Function	Conversion ID	Severity code	Meaning
	m0520	E	Unable to convert MOVE statement (moving special register or intrinsic function to national data item)
	m0521	E	Unable to convert MOVE statement (When there are multiple receiving items and they have different attributes)
	m0522	I	Convert MOVE statement (moving from numeric literal to national data item)
INITIALIZE statement	m0539	I	Convert INITIALIZE statement (When JAPANESE is specified in the REPLACING phrase of the INITIALIZE statement)

Table A.5 Source Text Operation

Function	Conversion ID	Severity code	Meaning
COPY statement	m0601	I	Expand COPY statement
	m0602	W	Convert REPLACING phrase of COPY statement
REPLACE statement	m0607	W	Convert REPLACE statement scope

A.2 Conversion Item Details

The following explains each conversion ID.

- Overview

  Explains the overview on conversion content.

- Third-party COBOL

  Shows an example of description in Third-party COBOL.

- NetCOBOL

  Shows a description example converted to syntax that can be compiled by NetCOBOL.

- Conversion Message

  Displays when the severity code is more than W.

- Supplementary Explanation

  Explains when there is a supplementary explanation.

In the following, unless otherwise noted, description examples and conversion examples are described in the fixed format reference format.

m0101 (Determine the condition of the $statement ($IF, $ELSE, $END) and convert it to comment line)

Overview

This is based on the condition statement in the source program and shown in the following $ statements. Convert selected parts of the source text to comment lines.

- $IF statement
  - Format 1: $IF constant-name [NOT] {< | > | =} value
  - Format 2: $IF constant-name [NOT] DEFINED
  - Format 3: $IF directive-setting SET

- $ELSE statement
- $END statement

This describes a "Conditional compilation."

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      $IF A DEFINED
          DISPLAY "A is defined".
      $ELSE
          DISPLAY "A is not defined".
      $END
```

NetCOBOL

When constant A is defined

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      *IF A DEFINED
            DISPLAY "A is defined".
      *ELSE
      *     DISPLAY "A is not defined".
      *END
```

When constant A is not defined

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      *IF A DEFINED
      *     DISPLAY "A is defined".
      *ELSE
           DISPLAY "A is not defined".
      *END
```

Supplementary Explanation

- Specify the constant value of the conditional compilation in the conversion information file. The conversion option to be specified for the conversion information file is determined by the $IF statement.

  Specification format varies depending on format of $IF statement.

  - Format 1 and format 2: Specify the conversion option CONSTANT
  - Format 3: Specify the conversion option DIRECTIVE

  For details, refer to "CONSTANT" and "DIRECTIVE" of "A.4 Specification Information of Conversion Information File."

- The following source program descriptions are not valid when there is a conditional compilation for the pre-compiler source conversion function to handle.

  - Named literal (The named literal is defined by level number 78)
  - CONSTANT directive of $SET statement

  In this case, specify this in the conversion information file instead.

  In addition, correct any statements in the source program that may update the constant-value.

m0102 (Output text data as conversion message when $DISPLAY statement is specified)

Overview

Converts the $DISPLAY statement to a comment line. Outputs the text data specified in $DISPLAY statement as the conversion message PRCV-m0102I.

However, $DISPLAY statements in scopes excluded by the condition are excluded.

Note

For [Winx64], the conversion message [PRCV-m0102I] is output to standard output.

For [Linux64], the conversion message [PRCV-m0102I] is output to standard error output.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      $DISPLAY HELLO WORLD
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      *DISPLAY HELLO WORLD
```

Conversion Message

When using compilation command

```
JMN0505I-I PRCV-m0102I HELLO WORLD
```

When using conversion command

```
PRCV-m0102I HELLO WORLD
```

Supplementary Explanation

When using compilation command.

The text data in $DISPLAY statement is displayed as an embedded character string of diagnostic message JMN0505I-I.

m0105 (Convert the $SET statement to comment line)

Overview

If $SET statement is described in source program, a conversion message is output with severity code W and the $SET statement is converted to a comment line.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      $SET FOLD-COPY-NAME"LOWER"     <-  PRCV-m0105W
       IDENTIFICATION DIVISION.
       PROGRAM-ID. A.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
```

```
      *SET FOLD-COPY-NAME"LOWER"
       IDENTIFICATION DIVISION.
       PROGRAM-ID. A.
```

Conversion Message

```
PRCV-m0105W  The compiler directive specified by $SET statement is invalid.
```

Supplementary Explanation

The compiler directive is not valid. Please check the compatibility of the NetCOBOL compiler option and the Micro Focus compiler directive.

For compatibility, refer to "A.3 Compiler Directive of Third-party COBOL."

m0107 (Convert to NetCOBOL non-reserved words. (Specify "-RV" option))

Overview

Convert words that are considered reserved words in NetCOBOL and not considered reserved words in Third-party COBOL. Suffix "-RV" at the end of corresponding word.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 RESUME PIC 9.

           MOVE 0 TO RESUME.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 RESUME-RV PIC 9.

           MOVE 0 TO RESUME-RV.
```

Supplementary Explanation

- There is no check for same name at the conversion.
- Words in the following areas are not converted.
  - Program name (PROGRAM-ID paragraph)
  - Library name (COPY statement)
- When a third-party COBOL reserved word is changed, a user-defined word may not be converted and may cause a NetCOBOL compile error. In this case, you can customize the third-party COBOL reserved words file to convert the word. For details, refer to "Appendix C How to Customize Reserved Words."
- NetCOBOL reserved words are:
  - Words specified by compiler option RSV (the type of a reserved word)
  - Words mentioned in "Synonym Compatibility Mode" of "Micro Focus Native Functions" in "NetCOBOL Language Reference"

  In the conversion result file, include the following information for the convert line:
  - Conversion ID
  - The compiler option RSV value to suppress conversion

  Output Example:

```
** conversion item information **
Line number conversion-ID Severity-code Detailed-information
xx          m0107         I             RESUME RSV(V90)
```

In the above example, "RESUME" is not treated as a NetCOBOL reserved word when specified as compiler option RSV (V90). To suppress conversion of reserved words, specify RSV value.

m0113 (Convert the line to a comment line when the first column is "*", X"00" to X"1F")

Overview

When "*" or X"00" to X"1F" are described in first column, convert the line to a comment line.

However, tab (X"09"), return (X"0D"), and line feed (X"0A") are excluded.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
*01000     MOVE  A  TO  B.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      *01000    MOVE  A  TO  B.
```

Supplementary Explanation

- Unless the indicator area is "*".
- Unless there are no characters in column 7 or later.

m0118 (Convert lines longer than 251 bytes)

Overview

For the variable format or free format, maximum line length of source program for third-party COBOL is 255 bytes. However, the maximum line length for NetCOBOL is 251 bytes. Therefore, a conversion message is output for lines that exceed the maximum length for NetCOBOL.

In this case, please wrap to fit within 251 bytes.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8-       ...         ----255
       PROCEDURE DIVISION.
           DISPLAY DISPLAY-DATA-X001 DISPLAY-DATA-X002  ...  DISPLAY-DATA-X019 DISPLAY-DATA-X00020.
            ^
            +----PRCV-m0118E
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8-     ...         ----255
       PROCEDURE DIVISION.
           DISPLAY DISPLAY-DATA-X001 DISPLAY-DATA-X002  ...  DISPLAY-DATA-X019 DISPLAY-DATA-X00020.
```

Conversion Message

```
PRCV-m0118E   A line that exceeds maximum length (251 bytes) of variable format or free
format is shown.
```

m0305 (Convert SWITCH 0 to 7)

Overview

Convert function name (insert "-"):

- From "SWITCH 0" to "SWITCH-0"
- From "SWITCH 1" to "SWITCH-1"
- From "SWITCH 2" to "SWITCH-2"
- From "SWITCH 3" to "SWITCH-3"
- From "SWITCH 4" to "SWITCH-4"
- From "SWITCH 5" to "SWITCH-5"
- From "SWITCH 6" to "SWITCH-6"
- From "SWITCH 7" to "SWITCH-7"

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
           SWITCH 0  IS  SW0
           SWITCH 7  IS  SW7
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
           SWITCH-0   IS  SW0
           SWITCH-7   IS  SW7
```

m0306 (Convert SWITCH 8)

Overview

Convert function name from "SWITCH 8" to "SWITCH-8" (insert "-").

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
           SWITCH 8  IS  SW8      <-  PRCV-m0306W
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
           SWITCH-8  IS  SW8
```

Conversion Message

```
PRCV-m0306W  SWITCH-8 is synonymous with SWITCH-0. It means same external switch.
```

Supplementary Explanation

In NetCOBOL, since SWITCH-8 has same meaning as SWITCH-0, a conversion message is output with severity code W.

m0307 (Convert NUMERIC SIGN IS TRAILING SEPARATE clause)

Overview

When NUMERIC SIGN clause is described in a SPECIAL-NAMES paragraph, inserts "SIGN IS TRAILING SEPARATE" into elementary item (*) of data description entry.

*: The elementary item is a signed numeric item without SIGN clause specified, and its usage is DISPLAY

In SPECIAL-NAMES, the NUMERIC SIGN clause is converted to a comment line.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
           NUMERIC  SIGN  IS  TRAILING  SEPARATE.

       01  G.
         02  A   PIC   9(4).
         02  B   PIC  S9(4)  BINARY.
         02  C   PIC  S9(4)  SIGN LEADING.
         02  D   PIC  S9(4).
         02  E   PIC  S9(4)  DISPLAY.

       77  F   PIC  S9(4).
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       SPECIAL-NAMES.
      *    NUMERIC  SIGN  IS  TRAILING  SEPARATE.

       01  G.
         02  A   PIC   9(4).
         02  B   PIC  S9(4)  BINARY.
         02  C   PIC  S9(4)  SIGN LEADING.
         02  D   PIC  S9(4)  SIGN IS TRAILING SEPARATE.
         02  E   PIC  S9(4)  DISPLAY  SIGN IS TRAILING SEPARATE.

       77  F   PIC  S9(4)  SIGN IS TRAILING SEPARATE.
```

Supplementary Explanation

When SIGN clause is in a group item, it also applies to elementary items of the group item.

m0310 (Delete EXTERNAL or DYNAMIC phrase of ASSIGN clause)

Overview

When EXTERNAL phrase or DYNAMIC phrase is used in the ASSIGN clause, the keyword is converted to a comment line.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
           SELECT  FILE1  ASSIGN  TO  EXTERNAL  FILE01.
           SELECT  FILE2  ASSIGN  TO  DYNAMIC   FILE02.

       01 FILE02 PIC X(100) VALUE "test.txt".
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

           SELECT  FILE1  ASSIGN  TO
      *                               EXTERNAL
                                              FILE01.
           SELECT  FILE2  ASSIGN  TO
      *                               DYNAMIC
                                              FILE02.

       01 FILE02 PIC X(100) VALUE "test.txt".
```

Supplementary Explanation

- In the following cases, performs concurrent conversion ID at the same time as this conversion.

ASSIGN clause	Condition	Concurrent conversion ID
DYNAMIC	No data item is declared for the file identifier	m0311

- When EXTERNAL is described in ASSIGN clause, set the following environment variable at program execution time.

  Environment variable name: File-identifier

  Environment variable value: The name of the input /output file

  When a data item is declared with same name as the file identifier (*1), a conversion error is raised to prevent input-output processing on file, where file name is set to a data item. In this case, rename either file identifier or data item and then execute conversion process again.

  *1: Including when a data item is created by m0311

m0311 (Create ASSIGN clause data name area for dynamic file allocation)

Overview

When "data name" is specified as file identifier in the ASSIGN clause and if that data name is not declared, create the data item in working-storage section.

If there is no working-storage section, the working-storage section is created.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

           SELECT  FILE1  ASSIGN  TO  FILENAME.

       FD  FILE1.
       01  REC1    PIC X(80).

       WORKING-STORAGE SECTION.
       *> "FILENAME" is not declared in working-storage section.

           MOVE  "FILE1.dat"  TO  FILENAME.    *> "FILENAME" is referenced
           OPEN  INPUT  FILE1.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

           SELECT  FILE1  ASSIGN  TO  FILENAME.

       FD  FILE1.
```

```
       01  REC1    PIC X(80).

       WORKING-STORAGE SECTION.
       01  FILENAME  PIC X(255).

           MOVE "FILE1.dat" TO FILENAME.  *> "FILENAME" is referenced
           OPEN  INPUT  FILE1.
```

Supplementary Explanation

- Create a data item with local attributes in the source program where the corresponding ASSIGN clause is declared.
- For Third-party COBOL, when a data name in program is not explicitly declared, the COBOL system implicitly creates it as an alphanumeric data item (area length is 255 bytes).

  For NetCOBOL, when a data name in the program is not explicitly declared, the NetCOBOL runtime system recognizes it as an external reference and writes an undefined error.
- In the following case, m0310 is also executed at the same time:
  - DYNAMIC is specified in ASSIGN clause.
  - A data item with the same name as file identifier is not declared.
- If neither DYNAMIC nor EXTERNAL is specified in the ASSIGN clause, it will be regarded as data name specification, and if this data name is not declared in the program, it will be subject to this conversion. You can change the default to be considered as a file identifier specification. For details, refer to "ASSIGN" of "A.4 Specification Information of Conversion Information File."

m0321 (Convert split key phrase (indexed organization) of RECORD KEY clause)

Overview

In the RECORD KEY clause and ALTERNATE RECORD KEY clause of an indexed file, when a split key is specified, it is converted.

Further it is also converted if it is referenced in START statements and READ statements.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
           SELECT  IXFILE  ASSIGN TO IXFILE2
                           ORGANIZATION IS INDEXED
                           ACCESS MODE  IS DYNAMIC
                           RECORD KEY   IS RKEY1=K1  K2
                           ALTERNATE RECORD KEY  IS RKEY2=A1  A2.

           START  IXFILE  KEY IS = RKEY1.

           READ   IXFILE  KEY IS RKEY2.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
           SELECT  IXFILE  ASSIGN  TO  IXFILE2
                           ORGANIZATION IS INDEXED
                           ACCESS MODE  IS DYNAMIC
      *                    RECORD KEY IS RKEY1=K1 K2
                           RECORD KEY IS       K1 K2
      *                    ALTERNATE RECORD KEY IS RKEY2=A1 A2.
                           ALTERNATE RECORD KEY IS       A1 A2.
      *     START IXFILE KEY IS = RKEY1.
            START IXFILE KEY IS =
```

```
                              K1 K2.
      *     READ IXFILE KEY IS RKEY2.

            READ IXFILE KEY IS
                              A1 A2.
```

m0402 (Convert named literal in report section to SYMBOLIC CONSTANT clause)

Overview

Moves the data description entry (named literal) level number 78 in the report section to the SYMBOLIC CONSTANT clause in SPECIAL-NAMES paragraph, then converts it.

If it cannot be moved, a conversion message is output with severity code E, and the data description entry level number 78 is unchanged.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       78  A   VALUE  1.
       78  B   VALUE  1000.
       78  C   VALUE  "ABC".
       78  D   VALUE  ALL ZERO.        <-  PRCV-m0402E
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
     SPECIAL-NAMES.
         SYMBOLIC CONSTANT
             A  IS  1
             B  IS  1000
             C  IS  "ABC"
             .

      *78  A   VALUE  1.
      *78  B   VALUE  1000.
      *78  C   VALUE  "ABC".
       78  D   VALUE  ALL ZERO.
```

Conversion Message

```
PRCV-m0402E The data description entry level number 78 cannot be moved to SYMBOLIC
CONSTANT clause.
```

Supplementary Explanation

- Convert the data description entry level number 78 to comment lines.
- When creating the SYMBOLIC CONSTANT clause, each division header, section header, and paragraph header will be created, if necessary.
- If the data description entry level number 78 is described in the library, a new library file is created (library file for SPECIAL-NAMES paragraph). The file name of library file for SPECIAL-NAMES paragraph is the original library file name with a one prefix character.

  To specify the prefix character, refer to "COPY-PREFIX-CHAR" of "A.4 Specification Information of Conversion Information File." Default prefix character is "S".

m0403 (Convert level-number 01 or 77 data items with OCCURS clause specified)

Overview

When OCCURS clause is specified in data description entry for level number 01 or 77, the FILLER with level-number 01 is generated, and original level number is incremented sequentially.

In the FILE SECTION, if OCCURS clause is specified in data description entry level number 01 or 77, a conversion message is output with severity code E.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       DATA DIVISION.
       FILE SECTION.
       FD FILE1.
       01  REC1 OCCURS 10.           <- PRCV-m0403E
         02 REC1-1 PIC X.
         02 REC1-2 PIC N.
       WORKING-STORAGE SECTION.
       01  G1  OCCURS  10.
         02  A   PIC  X.
         02  B.
           03  B1  PIC  X.
           03  B2  PIC  X.
         02  C   PIC X.

       77  G2  OCCURS  10  PIC  X.
           DISPLAY G1.
           DISPLAY G2.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       DATA DIVISION.
       FILE SECTION.
       FD FILE1.
       01  REC1 OCCURS 10.
         02 REC1-1 PIC X.
         02 REC1-2 PIC N.
       WORKING-STORAGE SECTION.
       01.
         02  G1  OCCURS  10.
           03  A   PIC  X.
           03  B.
             04  B1  PIC  X.
             04  B2  PIC  X.
           03  C   PIC X.
       01.
         02  G2  OCCURS  10  PIC  X.

           DISPLAY G1.
           DISPLAY G2.
```

Conversion Message

```
PRCV-m0403E   The OCCURS clause is specified for the File Section data description entry
level number 01 or 77.
```

Supplementary Explanation

- When level number is 49, this level number becomes 50 (In this case, a compile error occurs).

- When level number is 77, this level number becomes 02.
- When data description entry with OCCURS clause is referred without subscripting in the PROCEDURE DIVISION. In this case, add "1" as subscript.

```
Example: G1(1)
```

m0404 (Convert numeric items from COMP-X to BINARY)

Overview

Converts the numeric data item with USAGE IS COMP-X to USAGE IS BINARY. The number of digits does not convert.

When the number of digits exceeds 18 digits, a conversion message is output with severity code W.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1   PIC  9(1)  USAGE IS COMP-X.
       01  A2   PIC  9(18) USAGE IS COMP-X.
       01  A3   PIC  9(30) USAGE IS COMP-X.  <- PRCV-m0404W
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      *01  A1   PIC  9(1)  USAGE IS COMP-X.
       01  A1   PIC  9(1)  USAGE IS BINARY.
      *01  A2   PIC  9(18) USAGE IS COMP-X.
       01  A2   PIC  9(18) USAGE IS BINARY.
      *01  A3   PIC  9(30) USAGE IS COMP-X.
       01  A3   PIC  9(30) USAGE IS BINARY.
```

Conversion Message

```
PRCV-m0404W COMP-X is specified in the USAGE clause of numeric data items exceeding 18
digits.
```

Supplementary Explanation

- During compile, compiler option BINARY(BYTE) is enabled.
- When named literal is described with number of digits in the numeric data item, no conversion occurs.
- For numeric data item exceeding 18 digits, specify the compiler option ARITHMETIC (31). However, compiler option ARITHMETIC (31) and BINARY (BYTE) cannot be specified at the same time. You can consider either of the following as a work-around:

  a. Specify compiler option ARITHMETIC (31) and BINARY (WORD, MLBON) at the same time

  Note

  When BINARY(WORD) is specified, the size of the storage area allocated to a binary data item is determined by the number of digits specified in the PICTURE clause. In the program which considers area length of a binary item, operation of program may get affected. The number of digits is 27 and 28 digits, area length is not changed by compiler option BINARY.

  b. For the data item for which the conversion message PRCV-m0404W was output, correct the number of digits in data item to 18 digits

See

For details for compiler options ARITHMETIC and BINARY, refer to "NetCOBOL User's Guide."

For [Winx64]:

- "ARITHMETIC (Selects the operation mode)"
- "BINARY(binary data item handling)"

For [Linux64]:

- "ARITHMETIC (Specifies the operation mode)"
- "BINARY (binary data item handling)"

m0405 (Convert alphanumeric data items from COMP-X to BINARY)

Overview

Converts the alphanumeric data item with USAGE IS COMP-X to the numeric data item with USAGE IS BINARY. At this time, converts the number of digits in numeric data item into the maximum number of digits corresponding to the number of bytes in the alphanumeric data item. See the table below.

Number of bytes in alphanumeric data item	Number of digits in numeric data item
1 byte	2 digits
2 bytes	4 digits
3 bytes	7 digits
4 bytes	9 digits
5 bytes	12 digits
6 bytes	14 digits
7 bytes	16 digits
8 bytes	18 digits
9 to12 bytes	28 digits(*1)
13 bytes	31 digits(*1)
14 to 16 bytes	Cannot be converted (*2)

*1: Writes the conversion message with severity code W.

*2: When severity code is level E, no conversion occurs. Manual correction is required.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  B1   PIC  X(1)  USAGE IS COMP-X.
      01  B2   PIC  X(2)  USAGE IS COMP-X.
      01  B3   PIC  X(3)  USAGE IS COMP-X.
      01  B4   PIC  X(4)  USAGE IS COMP-X.
      01  B5   PIC  X(5)  USAGE IS COMP-X.
      01  B6   PIC  X(6)  USAGE IS COMP-X.
      01  B7   PIC  X(7)  USAGE IS COMP-X.
      01  B8   PIC  X(8)  USAGE IS COMP-X.
      01  B9   PIC  X(10) USAGE IS COMP-X. <- PRCV-m0405W
      01  B10  PIC  X(16) USAGE IS COMP-X. <- PRCV-m0405E
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

      *01  B1   PIC  X(1)  USAGE IS COMP-X.
       01  B1   PIC  9(02) USAGE IS BINARY.
      *01  B2   PIC  X(2)  USAGE IS COMP-X.
       01  B2   PIC  9(04) USAGE IS BINARY.
      *01  B3   PIC  X(3)  USAGE IS COMP-X.
       01  B3   PIC  9(07) USAGE IS BINARY.
      *01  B4   PIC  X(4)  USAGE IS COMP-X.
       01  B4   PIC  9(09) USAGE IS BINARY.
      *01  B5   PIC  X(5)  USAGE IS COMP-X.
       01  B5   PIC  9(12) USAGE IS BINARY.
      *01  B6   PIC  X(6)  USAGE IS COMP-X.
       01  B6   PIC  9(14) USAGE IS BINARY.
      *01  B7   PIC  X(7)  USAGE IS COMP-X.
       01  B7   PIC  9(16) USAGE IS BINARY.
      *01  B8   PIC  X(8)  USAGE IS COMP-X.
       01  B8   PIC  9(18) USAGE IS BINARY.
      *01  B9   PIC  X(10) USAGE IS COMP-X.
       01  B9   PIC  9(28) USAGE IS BINARY.
       01  B10  PIC  X(16) USAGE IS COMP-X.
```

Conversion Message

```
PRCV-m0405W COMP-X is specified in the USAGE clause of alphanumeric data items exceeding
8 bytes.
PRCV-m0405E COMP-X is specified in the USAGE clause of alphanumeric data items exceeding
8 bytes.
```

Supplementary Explanation

- During compile, compiler option BINARY(BYTE) is enabled.
- When named literal is described in number of digits of the alphanumeric data item, no conversion occurs.
- In any of the following cases, specify the compiler option ARITHMETIC (31).
  - Converted numeric data item exceeds 18 digits
  - A simultaneous conversion (m0509 or m0510) creates a numeric data item that exceeds 18 digits

  However, compiler option ARITHMETIC (31) and BINARY (BYTE) cannot be specified at the same time. You can consider either of the following as a work-around:

  a. Specify compiler option ARITHMETIC (31) and BINARY (WORD, MLBON) at the same time

  Note

  Area length of binary item is in word unit which is obtained from number of digits count. In programs which consider area length of binary items, operation of program may get affected. If alphanumeric data items before conversion is 12 bytes, compiler option BINARY has no effect on area length.

  b. For the data item for which the conversion message PRCV-m0405W was output, correct the number of digits in data item to 18 digits.

See

For details for compiler options ARITHMETIC and BINARY, refer to "NetCOBOL User's Guide."

For [Winx64]:

- "ARITHMETIC (Selects the operation mode)"
- "BINARY(binary data item handling)"

For [Linux64]:

- "ARITHMETIC (Specifies the operation mode)"
- "BINARY (binary data item handling)"

- When severity code is W, a data item is converted to a numeric data item with a larger number of bytes than before migration. In a program that is conscious of the data area length, the program operation may be affected.
- When severity code is I, a data item is converted to numeric data item with the same number of bytes as before migration. However, the number of digits may be different. In this case, the operation of size error condition may change.

m0406 (Convert from figurative constant ZERO or QUOTE to national nonnumeric literal)

Overview

When figurative constant ZERO or QUOTE is specified in VALUE clause of national data item, the figurative constant is converted to a national nonnumeric literal.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A    PIC  N(4)  VALUE  ZERO.
       01  B    PIC  N(4)  VALUE  ALL ZERO.
       01  C    PIC  N(4)  VALUE  QUOTE.
       01  D    PIC  N(4)  VALUE  ALL QUOTE.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A    PIC  N(4)  VALUE  ALL NC"O".
       01  B    PIC  N(4)  VALUE  ALL NC"O".
       01  C    PIC  N(4)  VALUE  ALL NC"“".
       01  D    PIC  N(4)  VALUE  ALL NC"“".
```

Supplementary Explanation

- The figurative constant ZERO includes ZEROES and ZEROS.
- The figurative constant QUOTE includes QUOTES.

m0407 (Convert from nonnumeric literal to national nonnumeric literal)

Overview

When nonnumeric literal is specified in VALUE clause of national data item, the nonnumeric literal is converted to a national nonnumeric literal.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A    PIC  N(7)  VALUE  "ＦＵＪＩＴＳＵ".
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A    PIC  N(7)  VALUE  NC"ＦＵＪＩＴＳＵ".
```

Supplementary Explanation

- When the nonnumeric literal contains half size character, if national data item encoding is SJIS, a compile error is output.
- When the continuation of the nonnumeric literal is used, a conversion message with severity code E is output.

```
PRCV-m0407E Nonnumeric literal with continuation that is specified in VALUE clause of
national data item cannot be converted.
```

- When the nonnumeric literal is a symbolic constant or a named literal, no conversion occurs.

m0509 (Convert relational conditions - comparing numeric data item and alphanumeric data item)

Overview

In conditional expressions, the following attributes can be converted:

- Comparison between packed decimal data items and alphanumeric data items (*)
- Comparison between binary data items and alphanumeric data items (*)
- Comparison between special register whose category is numeric and alphanumeric data items (*)

(*) Also converts alphabetic data items, alphanumeric edited data items and numeric edited data items. It is not converted when reference modification is to be performed on zoned decimal items and group items.

Conversion target is IF statement or EVALUATE statement. In this case, PERFORM statement with UNTIL phrase, or SEARCH statement, writes a conversion message with severity code E. No conversion occurs.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  A   PIC  X(4).
      01  B   PIC  9(4)  PACKED-DECIMAL.
      01  C   PIC  9(4)  BINARY.

          IF  A = B
          THEN
            DISPLAY  "*** OK ***"
          END-IF

          EVALUATE  TRUE
          WHEN  A = B
            DISPLAY  "*** OK ***"
          WHEN  A = C
            DISPLAY  "*** OK ***"
          WHEN  OTHER
            DISPLAY  "*** ERROR ***"
          END-EVALUATE

          PERFORM VARYING B FROM 1 BY 1 UNTIL B > A   <- PRCV-m0509E
            DISPLAY B
          END-PERFORM
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  A  PIC  X(4).
      01  B  PIC  9(4)  PACKED-DECIMAL.
      01  C  PIC  9(4)  BINARY.
      01  GEN--DATA--0001 IS GLOBAL  PIC 9(04).
```

```
       01  GEN--DATA--0002 IS GLOBAL  PIC 9(04).
       01  GEN--DATA--0003 IS GLOBAL  PIC 9(04).

      *    IF  A = B
           MOVE  B  TO  GEN--DATA--0001
           IF  A = GEN--DATA--0001
           THEN
             DISPLAY  "*** OK ***"
           END-IF

           MOVE  B  TO  GEN--DATA--0002
           MOVE  C  TO  GEN--DATA--0003
           EVALUATE  TRUE
      *    WHEN  A = B
           WHEN  A = GEN--DATA--0002
             DISPLAY  "*** OK ***"
      *    WHEN  A = C
           WHEN  A = GEN--DATA--0003
             DISPLAY  "*** OK ***"
           WHEN  OTHER
             DISPLAY  "*** ERROR ***"
           END-EVALUATE

           PERFORM VARYING B FROM 1 BY 1 UNTIL B > A
             DISPLAY B
           END-PERFORM
```

Conversion Message

```
PRCV-m0509E The comparison between alphanumeric data items specified in SEARCH statement
or PERFORM statement with UNTIL phrase and packed decimal items or binary item cannot be
converted.
```

Supplementary Explanation

- The binary item to be converted contains a data item with usage is BINARY, COMP, COMP-4, COMP-5, and COMP-X.

  However, in the alphanumeric data item with usage is COMP-X, if the number of digits satisfies the following, no conversion occurs.

  - Named literal is described with number of digits
  - The number of digits exceeds 14 (if the conversion ID m0405 results in severity code E)

- The packed decimal item to be converted contains a data item with usage is COMP-3 or PACKED-DECIMAL.
- In the binary item with usage is COMP-X, if this conversion and conversion ID m0405 are performed simultaneously, compiler option ARITHMETIC(31) may be required. For details, refer to "m0405."
- Generates the data name with the GLOBAL clause in the outermost program.
- The data name is generated for each conditional expression, and move statement (*1) is added immediately before evaluating the conditional expression. For this reason, multiple unnecessary move statements may be added by generating multiple data names for the same purpose.

  Unnecessary move statements can be deleted by using compiler option OPTIMIZE.

  *1: Move from binary data item or packed decimal data item to zoned decimal data item.

- You can specify the data name to be generated. For more information about how to specify, refer to "GENERATE-DATA-NAME" of "A.4 Specification Information of Conversion Information File."

  The serial number of nnnn (from 0001) is added to the end of the character string you specified. If you do not specify a string, the default name is "GEN--DATA--", and serial number "nnnn" is appended.

- Convert the following special registers.
  - RETURN-CODE
  - SORT-RETURN

m0510 (Convert relational conditions - comparing numeric data item and nonnumeric literal)

Overview

In conditional expressions, the following attributes can be converted:

- Comparison between packed decimal data item and nonnumeric literal
- Comparison between binary data item and nonnumeric literal
- Comparison between special register whose category is numeric and nonnumeric literal

Conversion target is IF statement or EVALUATE statement. In this case, PERFORM statement with UNTIL phrase, or SEARCH statement, writes a conversion message with severity code E. No conversion occurs.

Also converts a figurative constant.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

       01  A    PIC  9(4)  PACKED-DECIMAL.
       01  B1   PIC  9(4)  BINARY.
       01  B2   PIC  9(2)  BINARY.

           IF  A  =  "0123"
           THEN
             DISPLAY  "*** OK ***"
           END-IF

           EVALUATE  TRUE
           WHEN  B1 =  "0123"
             DISPLAY  "*** OK ***"
           WHEN  B2 =  "45"
             DISPLAY  "*** OK ***"
           WHEN  OTHER
             DISPLAY  "*** ERROR ***"
           END-EVALUATE

           PERFORM VARYING A FROM 1 BY 1 UNTIL A > "0123" <- PRCV-m0510E
             DISPLAY A
           END-PERFORM
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8

       01  A    PIC  9(4)  PACKED-DECIMAL.
       01  B1   PIC  9(4)  BINARY.
       01  B2   PIC  9(2)  BINARY.
       01  GEN--DATA-0001 IS GLOBAL  PIC 9(04).
       01  GEN--DATA-0002 IS GLOBAL  PIC 9(04).
       01  GEN--DATA-0003 IS GLOBAL  PIC 9(02).

      *    IF  A  =  "0123"
           MOVE  A  TO  GEN--DATA-0001
           IF  GEN--DATA-0001 = "0123"
           THEN
              DISPLAY  "*** OK ***"
```

```
           END-IF
           MOVE  B1  TO  GEN--DATA-0002
           MOVE  B2  TO  GEN--DATA-0003
           EVALUATE  TRUE
       *   WHEN  B1 =  "0123"
           WHEN  GEN--DATA-0002 = "0123"
             DISPLAY  "*** OK ***"
       *   WHEN  B2 =  "45"
           WHEN  GEN--DATA-0003 = "45"
             DISPLAY  "*** OK ***"
           WHEN  OTHER
             DISPLAY  "*** ERROR ***"
           END-EVALUATE

           PERFORM VARYING A FROM 1 BY 1 UNTIL A > "0123"
             DISPLAY A
           END-PERFORM
```

Conversion Message

```
PRCV-m0510E The comparison between nonnumeric literal specified in a SEARCH statement or
PERFORM statement with UNTIL phrase and packed decimal item or binary item cannot be
converted.
```

Supplementary Explanation

- The binary item to be converted contains a data item with usage is BINARY, COMP, COMP-4, COMP-5, and COMP-X.
- The packed decimal item to be converted contains a data item with usage is COMP-3 or PACKED-DECIMAL.
- In the binary item with usage is COMP-X, if this conversion and conversion ID m0405 are performed simultaneously, compiler option ARITHMETIC(31) may be required. For details, refer to "m0405."
- Generates the data name with GLOBAL clause in outermost program.
- The data name is generated for each conditional expression, and move statement (*1) is added immediately before evaluating the conditional expression. For this reason, multiple unnecessary move statements may be added by generating multiple data names for the same purpose.

  Unnecessary move statements can be deleted by using compiler option OPTIMIZE.

  *1: Move from binary data item or packed decimal data item to zoned decimal data item.
- You can specify the data name to be generated. For more information about how to specify, refer to "GENERATE-DATA-NAME" of "A.4 Specification Information of Conversion Information File."

  The serial number of nnnn (from 0001) is added to the end of the character string you specified. If you do not specify a string, the default name is "GEN--DATA--", and serial number "nnnn" is appended.
- When the nonnumeric literal is a symbolic constant or a named literal, no conversion occurs.
- Convert the following special registers.
  - RETURN-CODE
  - SORT-RETURN
- Convert the following figurative constants.
  - SPACE(S)
  - HIGH-VALUE(S)
  - LOW-VALUE(S)
  - QUOTE(S)

Note

The third-party COBOL allows the comparison between these figurative constants and packed decimal data items or binary data items. The value of the packed decimal data item or binary data item moved to the zoned decimal data item with the same number of digits is compared to these figurative constants.

For example, if a group item containing packed decimal data item or binary data item is initialized with a figurative constant, and then an elementary item is compared with this figurative constant, it is always false.

[Example]

```
      01 G01.
       02 G01-1 PIC S9(9) COMP-5.
       02 G01-2 PIC X(10).
      PROCEDURE DIVISION.
          MOVE LOW-VALUE TO G01.
          IF  G01-1  = LOW-VALUE     *> This comparison is always false.
          THEN
             DISPLAY "*** OK ***"
          END-IF.
```

This conversion will convert to a pre-migration compatible operation, which may be incorrect. If necessary, modify the source program.

m0511 (Convert relational conditions - comparing national data item and nonnumeric literal)

Overview

When there is a comparison between a national data item and nonnumeric literal, the national data item is converted by using the FUNCTION CAST-ALPHANUMERIC call.

If the two operands have different lengths, a conversion message with severity code E is output. No conversion occurs.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  A1    PIC  N(3).

          IF  A1  =  "ABCDEF"
          THEN
            DISPLAY  "***  OK ***"
          END-IF
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(3).

     *    IF  A1  =  "ABCDEF"
          IF  FUNCTION CAST-ALPHANUMERIC(A1)  =  "ABCDEF"
          THEN
            DISPLAY  "***  OK ***"
          END-IF
```

Conversion Message

```
PRCV-m0511E The comparison between a national data item and nonnumeric literal with
different lengths cannot be converted.
```

Supplementary Explanation

- When the nonnumeric literal is a symbolic constant or a named literal, does not convert.

- In the third-party COBOL, when writing a comparison that includes national data item, if the lengths of the two operands are different, a national space is assumed to exist on the right side of the shorter operand until it equals the length of the longer operand. In this conversion, national data items convert to CAST-ALPHANUMERIC function calls, so that an alphabetic space is assumed to exist. Therefore, if the lengths are different, no conversion is performed.
- If reference modification is to be performed on national data items, no conversion is performed.
- If a named literal is used in the PICTURE character-string of national data item, no conversion is performed.
- If the ALL literal is described, the lengths are assumed to be equal and are converted.
- The SEARCH ALL statement is also converted. However, if a national data item is specified as a key item in a table element, it converts the key item to a CAST-ALPHANUMERIC function call, resulting in a compilation error. These programs should be modified manually.

m0512 (Convert relational conditions - comparing national data item and alphanumeric data item)

Overview

When comparing the following, a national data item using the FUNCTION CAST-ALPHANUMERIC call.

- Comparison between national data item and alphanumeric data item (*)
- Comparison between national data item and zoned decimal item

(*) Also converts alphabetic data items, alphanumeric edited data items and numeric edited data items. It is not converted when reference modification is to be performed on group items.

If the two operands have different lengths, a conversion message with severity code E is output. No conversion occurs.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  N(3).
       01  A2    PIC  X(6).
        IF  A1  =  A2
           THEN
             DISPLAY  "***  OK ***"
           END-IF
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  N(3).
       01  A2    PIC  X(6).

      *    IF  A1  =  A2
           IF  FUNCTION CAST-ALPHANUMERIC(A1)  =  A2
           THEN
             DISPLAY  "***  OK ***"
           END-IF
```

Conversion Message

```
PRCV-m0512E The comparison between a national data item and alphanumeric data item or
zoned decimal data item with different lengths cannot be converted
```

Supplementary Explanation

- In the third-party COBOL, when writing a comparison that includes national data item, if the lengths of the two operands are different, a national space is assumed to exist on the right side of the shorter operand until it equals the

length of the longer operand. In this conversion, national data items convert to CAST-ALPHANUMERIC function calls, so that an alphabetic space is assumed to exist. Therefore, if the lengths are different, no conversion is performed.

- If reference modification is to be performed on either of two operand, no conversion is performed.
- If a named literal is used in PICTURE character-string of either or two operand, no conversion is performed.
- The SEARCH ALL statement is also converted. However, if a national data item is specified as a key item in a table element, it converts the key item to a CAST-ALPHANUMERIC function call, resulting in a compilation error. These programs should be modified manually.

m0513 (Convert relational conditions - comparing national data item and figurative constant ZERO or QUOTE)

Overview

When comparing the following, the figurative constant ZERO or QUOTE is converted to a national nonnumeric literal.

- Comparison between national data item and figurative constant ZERO
- Comparison between national data item and figurative constant QUOTE

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 A1 PIC N(8).

           IF A1 = ZERO
           THEN
             DISPLAY "***  OK  ***"
           END-IF

           IF A1 = QUOTE
           THEN
             DISPLAY "***  OK  ***"
           END-IF
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 A1 PIC N(8).

           IF A1 = ALL NC"O"
           THEN
             DISPLAY "***  OK  ***"
           END-IF

           IF A1 = ALL NC"“"
           THEN
             DISPLAY "***  OK  ***"
           END-IF
```

Supplementary Explanation

- The figurative constant ZERO includes ZEROES and ZEROS.
- The figurative constant QUOTE includes QUOTES.

m0514 (Convert MOVE statement - moving from alphanumeric data item to national data item)

Overview

When moving the following, the sending data item is converted using the FUNCTION NATIONAL call.

- Moving from alphanumeric data item to national data item (*)

(*) Also converts alphabetic data items, alphanumeric edited data items and numeric edited data items. It is not converted when reference modification is to be performed on group items.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  X(6) VALUE "TEST01".
       01  A2    PIC  N(6).

           MOVE  A1  TO  A2.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  X(6) VALUE "TEST01".
       01  A2    PIC  N(6).

      *    MOVE  A1  TO  A2.
           MOVE  FUNCTION NATIONAL(A1)  TO  A2.
```

m0515 (Convert MOVE statement - moving from zoned decimal data item to national data item)

Overview

When moving the following, the zoned decimal item is converted using the FUNCTION NATIONAL call.

- Moving from zoned decimal item to national data item

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  9(6) VALUE 123456.
       01  A2    PIC  N(6).

           MOVE  A1  TO  A2.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1    PIC  9(6) VALUE 123456.
       01  A2    PIC  N(6).

      *    MOVE  A1  TO  A2.
           MOVE  FUNCTION NATIONAL(A1)  TO  A2.
```

m0516 (Convert MOVE statement - moving from nonnumeric literal to national data item)

Overview

When moving from nonnumeric literal to national data item, add "NC" to nonnumeric literal, the half-size nonnumeric literal is converted to a full-size literal.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
```

```
       01  A1  PIC  N(8).

           MOVE  "ABC-ＪＰＮ"  TO  A1.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(8).

      *    MOVE  "ABC-ＪＰＮ"  TO  A1.
           MOVE   NC"ＡＢＣ－ＪＰＮ"  TO  A1.
```

Supplementary Explanation

- When a nonnumeric literal is continued, a conversion message with severity code E is output.

```
PRCV-m0516E Move from a continued nonnumeric literal to national data item cannot be
converted.
```

- When the nonnumeric literal is a symbolic constant or a named literal, it is not converted.

m0517 (Convert MOVE statement - moving from hexadecimal nonnumeric literal to national data item)

Overview

When moving from hexadecimal nonnumeric literal to national data item, the hexadecimal nonnumeric literal is converted to a national hexadecimal nonnumeric literal.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(3).

            MOVE  X"816081628163"  TO  A1.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(3).

      *     MOVE  X"816081628163"  TO  A1.
            MOVE  NX"816081628163" TO  A1.
```

Supplementary Explanation

- When the specifying a continued hexadecimal nonnumeric literal, a conversion message is output with severity code E.

```
PRCV-m0517E Move from hexadecimal nonnumeric literal with continuation to national
data item cannot be converted.
```

- When the nonnumeric literal is a symbolic constant or a named literal, it is not converted.
- If the character constant contains half size characters, a compile error occurs if the encoding of the national data item is SJIS.

m0518 (Convert MOVE statement - moving from national data items to alphanumeric data item)

Overview

When moving the following, a national data item is converted to the FUNCTION CAST-ALPHANUMERIC call.

- Moving from national data item to alphanumeric data item

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(3)  VALUE "ＡＢＣ".
       01  A2  PIC  X(6).

           MOVE  A1  TO  A2.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(3)  VALUE NC"ＡＢＣ".
       01  A2  PIC  X(6).

      *    MOVE  A1  TO  A2.
           MOVE  FUNCTION CAST-ALPHANUMERIC(A1) TO A2.
```

m0519 (Convert MOVE statement - moving from figurative constant ZERO or QUOTE to national nonnumeric literal)

Overview

When moving from figurative constant ZERO or QUOTE to national data item, converts figurative constant ZERO or QUOTE to national nonnumeric literal.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(8).

            MOVE  ZERO       TO  A1.
            MOVE  ALL ZERO   TO  A1.
            MOVE  QUOTE      TO  A1.
            MOVE  ALL QUOTE  TO  A1.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(8).

            MOVE  ALL NC"Ｏ" TO  A1.
            MOVE  ALL NC"Ｏ" TO  A1.
            MOVE  ALL NC"“" TO  A1.
            MOVE  ALL NC"“" TO  A1.
```

Supplementary Explanation

- The figurative constant ZERO includes ZEROES and ZEROS.
- The figurative constant QUOTE includes QUOTES.

m0520 (Unable to convert MOVE statement (moving special register or intrinsic function to national data item))

Overview

When moving from special register or intrinsic function to national data item, a conversion message is output with severity code E. No conversion occurs.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  X(8).
       01  A2  PIC  N(8).

            MOVE  CURRENT-DATE  TO  A2.              <- PRCV-m0520E
            MOVE  FUNCTION UPPER-CASE (A1)  TO  A2.   <- PRCV-m0520E
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  X(8).
       01  A2  PIC  N(8).

            MOVE  CURRENT-DATE  TO  A2.
            MOVE  FUNCTION UPPER-CASE (A1)  TO  A2.
```

Conversion message

```
PRCV-m0520E MOVE from special register or intrinsic function to national data item
cannot be converted.
PRCV-m0520E MOVE from special register or intrinsic function to national data item
cannot be converted.
```

m0521 (Unable to convert MOVE statement - when multiple data items on the receiving side have different attributes)

Overview

When multiple data items on the receiving side are used and they have different attributes, a conversion message is output with severity code E. No conversion occurs.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(8).
       01  A2  PIC  X(8).
       01  A3  PIC  N(8).

            MOVE  ZERO  TO  A1  A2  A3.      <- PRCV-m0521E
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01  A1  PIC  N(8).
       01  A2  PIC  X(8).
       01  A3  PIC  N(8).

            MOVE  ZERO  TO  A1  A2  A3.
```

Conversion Message

```
PRCV-m0521E Moving to multiple receiving items with different attributes cannot be
converted.
```

m0522 (Convert MOVE statement - moving from numeric literal to national data item)

Overview

The character-string that excludes the nonnumeric character from the numeric literal is converted to the national character-string, and enclosed with NC" and a quotation mark.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  A1  PIC  N(8).

           MOVE  1           TO  A1.
           MOVE  1.23        TO  A1.
           MOVE  +1000       TO  A1.
           MOVE  -0.456      TO  A1.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      01  A1  PIC  N(8).

           MOVE  NC"１"       TO  A1.
           MOVE  NC"１２３"   TO  A1.
           MOVE  NC"１０００" TO  A1.
           MOVE  NC"０４５６" TO  A1.
```

Supplementary Explanation

When the numeric literal is a symbolic constant or a named literal, it is not converted.

m0539 (Convert INITIALIZE statement - when JAPANESE is specified in the REPLACING phrase of the INITIALIZE statement)

Overview

When "JAPANESE" is specified in REPLACING phrase of INITIALIZE statement, converts REPLACING phrase.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
          INITIALIZE  G1   REPLACING  JAPANESE  BY  "あ".
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
     *    INITIALIZE  G1   REPLACING  JAPANESE  BY  "あ".
          INITIALIZE  G1   REPLACING  NATIONAL  BY  NC"あ"
                                NATIONAL-EDITED  BY  NC"あ".
```

Supplementary Explanation

- In third-party COBOL, when "JAPANESE" is specified for REPLACING phrase, the category to be initialized is only in national language.
  In this conversion, also initializes national edited.

  Therefore, if the data item specified in the INITIALIZE statement or the data item subordinate to it contains a national edited item, the operation will be different from the behavior before migration.

  If you want to exclude national edited items from initialization, modify the pre-conversion source program by rewriting "JAPANESE" to "NATIONAL".

- If you code the following in the BY phrase following the "JAPANESE" in the REPLACING phrase, they will be converted.

BY phrase	After Conversion
Nonnumeric literal (Example: "あ")	National nonnumeric literal (Example: NC"あ")
Hexadecimal nonnumeric literal (Example: X"8160")	National hexadecimal nonnumeric literal (Example: NX"8160")
Figurative constant ZERO (Including ZEROES and ZEROS)	NC"0"
Figurative constant QUOTE (Including QUOTES)	NC"”"

- If the character constant contains half size characters, a compilation error occurs if the encoding of the national data item is SJIS.

- When the nonnumeric literal is a symbolic constant or a named literal, it is not converted.

m0546 (Unable to convert MOVE statement - In abbreviated combined relation conditions, when required to rewrite the left operand)

Overview

Combined relational conditions (multiple condition expressions) can be abbreviated if the left operand or the comparison operator are the same. This is called "abbreviated combined relational conditions."

The abbreviated conditional expression is also converted according to conversion rules.

However, when the left operand must be rewritten, no conversion occurs. A conversion message is output with severity code E.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 A PIC X(4).
       01 B PIC 9(4) PACKED-DECIMAL.
       01 C PIC X(4).

          IF  A  =  B  OR  SPACE
          THEN
            DISPLAY "OK"
          END-IF

          IF  B  =  C  OR  ZERO            <- PRCV-m0546E
          THEN
            DISPLAY "OK"
          END-IF
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 A PIC X(4).
       01 B PIC 9(4) PACKED-DECIMAL.
       01 C PIC X(4).
       01 GEN--DATA-001 PIC 9(4).
       01 GEN--DATA-002 PIC 9(4).

            MOVE B TO GEN--DATA--001
      *     IF  A  =  B  OR  SPACE
            IF  A  =  GEN--DATA--001  OR  SPACE
            THEN
              DISPLAY "OK"
            END-IF

            MOVE B TO GEN--DATA--002
            IF  B  =  C  OR  ZERO
            THEN
              DISPLAY "OK"
            END-IF
```

Conversion Message

```
PRCV-m0546E  The left operand in abbreviating combined relation conditions cannot be
converted.
```

Supplementary Explanation

- When a conversion (m0510 or m0509) to generate data is discontinued, unnecessary data name may result.
- When this conversion message is output, if the left operand or the comparison operator does not be abbreviated, it can be converted. After correcting the pre-conversion source program, convert again.

m0601 (Expand COPY statement)

Overview

When COPY statement is used, converted contents of read libraries are expanded in the source program. However, if the REPLACING phrase is specified in COPY statement, it is not expanded in the source program.

For the conversion of COPY with the REPLACING phrase, refer to conversion ID m0602.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
        COPY COPY001.
```

Content of library (COPY001)

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 DATA01 PIC X(100).
       01 DATA02 PIC 9(9) COMP-X.
```

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
      * COPY COPY001.
       01 DATA01 PIC X(100).
```

```
      *01 DATA02 PIC 9(9) COMP-X.
       01 DATA02 PIC 9(9) BINARY.
```

Supplementary Explanation

Using conversion command

Expands only when specified in conversion information file.

For how to specify the file, refer to "EXPAND-COPY" of "A.4 Specification Information of Conversion Information File."

m0602 (Convert COPY REPLACING phrase)

Overview

When COPY with REPLACING phrase is used, REPLACING is ignored during conversion processing. A conversion message is output with severity code W.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
        COPY COPY001 REPLACING ==SIZE== BY ==8==.      <- RCV-m0602W
```

Content of library text (COPY001)

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 IN-DATA PIC 9(SIZE) COMP-X.    <- (*1)

       01 OUT-DATA PIC 9(8)   COMP-X.    <- (*2)
```

*1: After REPLACING phrase, this description will be subject to conversion ID m0404.

*2: This description will be subject to conversion ID m0404.

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
        COPY COPY001 REPLACING ==SIZE== BY ==8==.
```

Content of library post conversion (COPY001)

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       01 IN-DATA PIC 9(SIZE) COMP-X.  <- (*3)
      *01 OUT-DATA PIC 9(8)   COMP-X.
       01 OUT-DATA PIC 9(8)   BINARY.
```

*3: This description is not subject to conversion ID m0404 because it is not replaced by REPLACING phrase.

Conversion Message

```
PRCV-m0602W COPY statement with REPLACING phrase is used. Conversion does not perform
the REPLACING phrase.
```

Supplementary Explanation

- REPLACING phrase is ignored during conversion. As a result, the conversion and the replacing may not be as intended during compilation. Check the results and correct manually as needed.

- Depending on the COPY statement format, library file allocation errors may occur during compilation. For the COPY statement format and corrective action, refer to "When COPY statement is not expanded" of "2.1.4.1.3 Handling of Library Text File ."

- Using compilation command

  The line-number of compiler message in the library text is the line-number of the post-conversion library text. Also, debugged program becomes a post-conversion library text.

m0607 (Convert REPLACE statement scope)

Overview

When REPLACE statement is described, REPLACE is ignored during conversion processing. A conversion message is output with severity code W.

Third-party COBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       REPLACE ==SIZE== BY ==8==.       <- RCV-m0607W
       01 IN-DATA PIC 9(SIZE) COMP-X    <-(*1)
       01 OUT-DATA PIC 9(8)   COMP-X.   <-(*2)

       REPLACE OFF.
```

*1: After the REPLACE statement is performed, this description will be subject to conversion ID m0404.

*2: This description will be subject to conversion ID m0404.

NetCOBOL

```
0--------1--------2--------3--------4--------5--------6--------7--------8
       REPLACE ==SIZE== BY ==8==.
       01 IN-DATA PIC 9(SIZE) COMP-X.    <-(*3)
      *01 OUT-DATA PIC 9(8)   COMP-X.
       01 OUT-DATA PIC 9(8)   BINARY.
       REPLACE OFF.
```

*3: This description is not subject to conversion ID m0404 because it is not replaced by REPLACE statement.

Conversion Message

```
PRCV-m0607W  REPLACE statement is described. Convert without performing the REPLACE
statement.
```

Supplementary Explanation

REPLACE is ignored during conversion. As a result, the conversion result and REPLACE statement may not be as intended during compilation. Check the results and correct manually as needed.

A.3 Compiler Directive of Third-party COBOL

This section explains the third-party COBOL compiler directives that are compatible with NetCOBOL.

When you are using these compiler directives, enable the required compiler options in NetCOBOL.

If you use a compiler directive which is not described here, it is necessary to investigate to see whether it is possible to achieve the same behavior post migration.

Compatibility

The meaning of the symbol in the table is given below.

A: Supported by a compiler option

B: Supported by a compiler option. However, there is no support depending on specified value or verification is required as there is a difference in program operation

C: Supported by a specification other than compiler option

D: Supported by a conversion option of the pre-compiler source conversion function

Compiler Directive of Third-party COBOL	Meaning	Compatibility	NetCOBOL
ANIM	Whether the debugging function and the COBOL Error Report should be used.	A	Compiler option TEST
{APOST|QUOTE}	Figurative constant QUOTE handling.	A	Compiler option { APOST | QUOTE}
ARITH	Definition of maximum number of digits of a numeric data item (COMPAT or EXTEND). Note: Specifiable values and default values are different.	B	Compiler option ARITHMETIC
ASSIGN	Specifies how to assign file name when neither EXTERNAL nor DYNAMIC is specified in the ASSIGN clause.	D	Conversion ID m0311
BOUND	Checks whether subscripts, indexes, and reference modifications are out of range.	A	Compiler option CHECK(BOUND)
CASE	Lowercase handling(Lowercase characters in the constants including the program name literal, the constants in CALL statement, CANCEL statement, ENTRY statement, and INVOKE statement, are distinguished from uppercase characters).	A	Compiler option ALPHAL(WORD)
CHECK	Whether the CHECK function should be used. Note: NetCOBOL has no function corresponding to LINKCHECK, PARAMCOUNTCHECK, and RECURSECHECK in Micro Focus COBOL.	B	Compiler option CHECK(BOUND,PRM)
CHECKNUM	Checks data exceptions (whether the value matching the attribute format is contained in the numeric data item and whether the divisor is zero).	B	Compiler option CHECK(NUMERIC)
COMP-5	Handle high order bit of an unsigned binary data item as a numeric value. Note: In NetCOBOL, COMP-5 and BINARY are also targeted.	B	Compiler option BINARY(WORD,MLBOFF)

Compiler Directive of Third-party COBOL	Meaning	Compatibility	NetCOBOL
CONSTANT	Define constants used in the program.	D	Conversion ID m0101
COPYEXT	Specify extension when file name in COPY statement is specified without an extension. Note: The default value of extension is different.	C	Environment variable COB_LIBSUFFIX
COPYLIST	Display a library text. Note: In NetCOBOL, segment-number cannot be specified.	B	Compiler option COPY
COPYPATH	Specify the list of directories for compiler to search for copy files.	C	W Environment variable COB_COBCOPY or -I option L Environment variable COBCOPY or -I option
CURRENCY-SIGN	Currency symbol handling. Note: NetCOBOL can specify only ￥or $.	B	Compiler option CURRENCY
DATAMAP	Whether data map listings, program control information listings, and section size listings should be written.	A	Compiler option MAP
DEFAULTBYTE	Initializing items without a VALUE phrase in WORKING-STORAGE. Note: In NetCOBOL, specified value is hexadecimal.	A	Compiler option INITVALUE
DIALECT	Match the check-time and runtime behavior with specified dialect. Note: Check each compiler directives.	-	-
DIRECTIVES	Specify the option file. Note: Specified format and specified value are different.	C	W Environment variable COB_OPTIONS L Environment variable COBOLOPTS or -i option
FLAG FLAGSTD	Whether messages should be output depending on the difference between standards. Note: The values that can be specified are different.	B	Compiler option CONF

Compiler Directive of Third-party COBOL	Meaning	Compatibility	NetCOBOL
L FOLD-COPY-NAME	Determines whether library file names should be converted to upper case or lower case.	C	Environment variable COB_COPYNAME
FORM	Number of line per page of compiler listing. Note: The values that can be specified are different.	B	Compiler option LINECOUNT
IBMCOMP	The elementary item size of binary data is treated as the area length of the word unit obtained from the number of digits and the high order end bit is treated as a numeric value of unsigned binary item.	A	Compiler option BINARY(WORD,MLBOFF)
LIST LISTPATH	Whether compiler listing should be written and the output destination specified.	A	W Compiler option PRINT or -P option L -P option
LISTWIDTH LW	Number of characters per line in the compiler listing. Note: The values that can be specified are different.	B	Compiler option LINESIZE
LITLINK	Program structure specification. Note: The values that can be specified and the default value are different.	B	Compiler option DLOAD
MF MFLEVEL	Enable Micro Focus specific reserved words.	B	Compiler option RSV
NSYMBOL	Specify encoding for national data items. Note: In NetCOBOL, specifies encoding of alphanumeric data items and national data items. In addition, there is a limit to the combinations that can be specified.	B	Compiler option ENCODE
OPT	Global optimization handling. Note: In NetCOBOL, optimization level cannot be specified.	B	Compiler option OPTIMIZE
PRINT	Whether compiler listing should be written and the output destination specified.	A	W Compiler option PRINT or -P option L -P option
REENTRANT	Specifies multithread program creation. Note: The values that can be specified are different.	B	W Compiler option THREAD(MULTI)

Compiler Directive of Third-party COBOL	Meaning	Compatibility	NetCOBOL
			L Compiler option THREAD(MULTI) or -Tm option
RESEQ	Source program sequence number area specification. Note: The default value are different.	A	Compiler option NONUMBER
RUNTIME-ENCODING	Specify encoding for the runtime.	B	Compiler option ENCODE
SOURCE-ENCODING	Specify encoding for a source program.	B	Compiler option SCS
SOURCEFORMAT	Reference format type. Note: The default value are different.	A	Compiler option SRF
TRUNC	Whether the truncation function should be used. Note: The values that can be specified and the default value are different.	B	Compiler option TRUNC
UNICODE	Specify endian for national data items. Note: In NetCOBOL, specifies encoding of alphanumeric data items and national data items. In addition, there is a limit to the combinations that can be specified.	B	Compiler option ENCODE
WARNING	Diagnostic message level. Note: The default value are different.	A	Compiler option FLAG
XREF	Whether a cross-reference list should be output.	A	Compiler option XREF

Note

In third-party COBOL, compiler directives can be described in $IF statement.

```
$IF directive-setting SET
```

When $IF statement is described, it is required to specify conversion option "DIRECTIVE" in conversion information file.

For details, refer to conversion ID m0101.

A.4 Specification Information of Conversion Information File

Conversion option	Specification value	Conversion ID	Description
ASSIGN	DYNAMIC or EXTERNAL	m0311	Specify how to assign a file name when neither DYNAMIC nor EXTERNAL is specified in the ASSIGN clause. - DYNAMIC: Assumes that a data name is specified and generates a data name if it is not defined. - EXTERNAL: Assumes that a file identifier is specified and does not generate a data name. The default value is DYNAMIC.
CONSTANT	constant-name(numeric) or constant-name "character-string"	m0101	Specify constants of conditional compilation for [Format 1] or [Format 2]. - When specified value as numeric, enclose it in parentheses, and when specified value as an alphanumeric character, enclose it in quotation marks. - When specifying multiple values, specify followed by a comma. Example: `CONSTANT=A(1),B"123"` **Rules of constant-name** The constant-name must be a character string and number of characters permitted as the user-defined word in NetCOBOL. However, it should not contain national characters. **Rules of value** - Numeric must be 1 to 31 digits. Also, all digits must be numeric. - Alphanumeric characters must be character string and number of characters permitted as the nonnumeric literal in NetCOBOL. **Notes** - Lowercase letters in the constant-name are treated as equivalent to uppercase letters. - This specification is applicable only for the condition determination of conditional compilation. When referencing in named literal, manual correction is required separately.

Conversion option	Specification value	Conversion ID	Description
			- There should be no space before or after parentheses, quotes, and commas.
COPY-PREFIX-CHAR	character	m0403	Specify the prefix character of the library file name for SPECIAL-NAMES paragraph. The library file for SPECIAL-NAMES paragraph is created when the data description entry of level 78 is specified in the library text. The file name of library file for the SPECIAL-NAMES paragraph is the original library file name with a one prefix character. Default prefix character is "S".
DIRECTIVE	character-string	m0101	Specify the character string of conditional compilation for [Format 3]. Specify multiples values following a comma. Example: `DIRECTIVE=P64,MF"10"` **Rules** - The character string must be 1 to 30 characters. - The character string must be ASCII characters. **Notes** - This specification is applicable only for the condition determination of conditional compilation. It does not affect interpretation during compilation. For the compiler directive compatibility, refer to "A.3 Compiler Directive of Third-party COBOL." - The character string must match the description in source program. - Lowercase letters in the character strings are treated as equivalent to uppercase letters. - There should be no space before or after commas.

Conversion option	Specification value	Conversion ID	Description
WORKDIR	directory-name	-	Specify the directory name of the work file used by the pre-compiler source conversion function. - Using compilation command It also writes the post-conversion source file that is created as an intermediate file to the same directory. The default directory is below: - Using compilation command .preconv directory under the directory where the pre-conversion source file is located. (.preconv directory is created automatically). - Using conversion command Directory specified by -o option.
GENERATE-DATA-NAME	character-string	m0509 m0510	Creates a data name with serial number "nnnn (nnnn: 0001 to 9999)" appended to the end of specified character string. Default data name is "GEN--DATA--nnnn." **Rules** - The character string must be 1 to 26 characters (Within 30 characters including the serial number 4 digits). - The character string must be a character string permitted as the user-defined word in NetCOBOL. However, it should not contain national characters.
EXPAND-COPY [Using conversion command]	YES or NO	m0601	When COPY statement is used, specify whether the library text is expanded in the post-conversion source program (YES) or create a file as post-conversion library file (NO). Default value is NO.
REMOVE-RSW	file-name-of-customized-reserved-words-file	m0107	In third-party COBOL, when a specific word is deleted from reserved words, specify file name of customized reserved words file. For how to customize, refer to "Appendix C How to Customize Reserved Words."

Conversion Information File Limits

When the description of the conversion information file exceeds the maximum allowed, the quantity that fits in this range is considered valid and conversion process is continued.

a. Specified value in conversion information file

W

Maximum number of bytes of specified value for one conversion option is 1,024 bytes (in UTF-8 representation).

L

Maximum number of bytes of specified value for one conversion option is 1,024 bytes.

b. Number of constant names specified in CONSTANT and the number of character strings specified in DIRECTIVE

Total number of the number of constant names specified in CONSTANT and the number of character strings specified in DIRECTIVE is a maximum of 100.

This value is valid in pre-conversion source program units.

Note

If you specify a value that does not conform to the rules, the specified option is ignored.

Appendix B Conversion Result File

This section explains the file written by the pre-compiler source conversion function.

The pre-compiler source conversion function writes the conversion result of the program to the conversion result file.

Format of conversion result file.

```
NetCOBOL Vxx.x.x                              WED JAN 16 13:30:55 2019

** CONVERSION INFORMATION **

  CONVERSION TYPE                          = m                    [1]
  CONVERSION INFORMATION FILE NAME         = A.ini                [2]

  CONVERSION OPTIONS      ESTABLISED VALUE                        [3]
  CONSTANT                A(2),B"ABC"
  DIRECTIVE               P64
    :

  COMPILER OPTIONS                                                [4]
  ENCODE(SJIS)
  RSV(V1100)
   :

** FILE INFORMATION **                                           [5]

  PRE-CONVERSION SOURCE PROGRAM            = A.cob
  POST-CONVERSION SOURCE PROGRAM           =.\out\A.cob

QUALIFIER        IN/OUT   FILE-NAME
[6]              [7]      [8]
      1           IN       COPY001.cbl
      1           OUT      ./out/COPY001.cbl

** CONVERSION ITEM INFORMAION **

LINE  CONV-ID SEVERITY   DETAIL INFORMATION
[9]   [10]    [11]       [12]
5     m0306   W           SWITCH-8 is synonymous with SWITCH-0.
                          It means same external switch.
11    m0404   I
1-1   m0404   I
13    m0101   I
15    m0101   I
17    m0101   I
18    m0107   I            RESUME RSV(V90)
20    m0107   I            RESUME RSV(V90)
21    m0107   I            RESUME RSV(V90)

** CONVERSION RESULT **
  CPU TIME               =  1.550 SECONDS                       [13]
  HIGHEST SEVERITY CODE  = W                                    [14]
```

[1] CONVERSION TYPE

Indicates the conversion type specified by the -CV option.

[2] CONVERSION INFORMATION FILE NAME

Indicates the name of the conversion information file specified by the -CV option. If the conversion information file is not specified, it indicates a space.

[3] CONVERSION OPTIONS

Indicates a list of established conversion options.

[4] COMPILER OPTIONS

Among the compiler options that are valid during conversion, indicates a list of established compiler options.

[5] FILE INFORMATION

Indicates the path names of the pre-conversion source program and post-conversion source program.

[6] QUALIFIER

A number that identifies the pre-conversion and post-conversion library file. It is allocated to COPY statement in ascending order from 1 in increments of 1.

[7] IN/OUT

Indicates the type of the file name.

- IN

  Library file copied from the pre-conversion source program.

- OUT

  Post-conversion library file. If the post-conversion library file is not created, this is blank.

[8] FILE-NAME

Indicates the path name.

[9] LINE

Indicates the line number of the pre-conversion source file in the following format:

```
[Qualifier-]line-number
```

[10] CONV-ID

The converted conversion ID is displayed.

[11] SEVERITY

Indicates the converted severity code.

[12] DETAIL INFORMATION

Indicates the detailed information when it exists for each conversion ID. When the severity code of the conversion ID is W or more, also displays a conversion message.

[13] CPU TIME

Indicates the time required for conversion.

[14] HIGHEST SEVERITY CODE

Indicates the highest severity code.

Appendix C How to Customize Reserved Words

This section explains about how to customize the reserved words in the pre-compiler source conversion function.

Reserved Words File

The product installation directory contains the following files:

For [Winx64]

File name	Explanation
F4AFCVM.TXT	Reserved words for third-party COBOL.

For [Linux64]

File name	Explanation
/opt/FJSVcbl64/config/JMNCVM.txt	Reserved words for third-party COBOL.

This file is a text file.

How to Customize

1. Copy the reserved words file from the installation directory of the product to any directory. The file name can be renamed to any name.
2. Edit the reserved words file. Refer to "How to Edit the Reserved Words File."
3. Specify the path name of the edited file in "REMOVE-RSW" of the conversion information file.

Note

Do not edit the reserved words file in the product installation directory directly.

How to Edit the Reserved Words File

The following shows how to edit the reserved words file:

1. Identify the words you want to remove from the reserved word.
2. Edit the reserved words file using the editor. Record is 54 bytes. (Record length is fixed.)

   The file format is shown below:

   Reserved Words File (Initial State)

```
#                                                 #
# comment line for explanation                    #
#                                                 #
Column number
1---+----10---+----20---+----30---+----40---+----50---
 001 ABSENT                        :    X      00
 002 ABSTRACT                      :           11
 003 ACCEPT                        :V XXXXXX  X01 XXXX
 004 ACCESS                        :  XXXXXX  X01 XXXX
 005 ACQUIRE                       :           07
 006 ACTIVE-CLASS                  :    X      12    X
      :
    (omission)
      :
 686 ZEROES                        :F XXXXXX  X01 XXXX
```

```
 687 ZEROS  <- Last reserved word  :F XXXXXX  X01 XXXX
#                                                    #
```

Explanation of columns

Column	Meaning	Explanation
1	Editable area	Specify "*" at this position to remove the word from the reserved words.
2 to 4	Serial number area	Indicates the serial number of reserved words. This number is used by the pre-compiler source conversion function.
6 to 35	Name of the reserved word	Arranged in ascending order with ASCII code.
36	Delimiter	This delimiter is used by the pre-compiler source conversion function.
37 to 54	Reserved words type	This information is used by the pre-compiler source conversion function.

Specify "*" in the first column of the word to be removed from the reserved words.

- Be careful not to change or update the column position.
- Do not change the contents of columns 2 to 54.
- It is not possible to add a word to the reserved words file.

3. When editing the file fails, please take a new copy from the original file to create a new edited file.

The edited reserved words file is shown below.

Reserved Words File (After Editing)

```
#                                                    #
# comment line for explanation.                      #
#                                                    #
Column number
1---+----10---+----20---+----30---+----40---+----50---
 001 ABSENT                        :    X      00
 002 ABSTRACT                      :           11
 003 ACCEPT                        :V XXXXXX  X01 XXXX
*004 ACCESS                        :  XXXXXX  X01 XXXX
 005 ACQUIRE                       :           07
 006 ACTIVE-CLASS                  :    X      12    X
*007 ACTUAL                        :     XX    00   X
       :
     (omission)
       :
 686 ZEROES                        :F XXXXXX  X01 XXXX
 687 ZEROS <-Last reserved word    :F XXXXXX  X01 XXXX
#                                                    #
```

This example shows "ACCESS" and "ACTUAL" are removed from the reserved words.

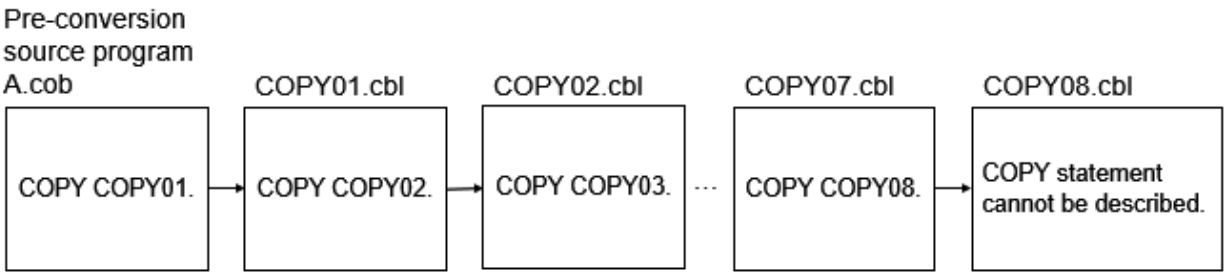
Appendix D Restrictions

This section explains the quantitative restrictions of the pre-compiler source conversion function.

The following values are based on pre-conversion source program unit. If there is a description exceeding this restriction, a message will be output and conversion process terminated.

a. Hierarchy level of COPY statement

Maximum hierarchy level (nested) of COPY statement is 8.



b. Number of data item

Maximum number of data description entries that can be defined in data division is 95,000.

c. Number of MOVE statements

Maximum number of MOVE statements that can be converted within the MOVE statement in procedure division is 3,200.

d. Number of conditional expressions

Maximum number of conditional expressions that can be converted within the conditional expressions in procedure division is 1,700. The maximum number of COBOL words constituting one operand of conditional expression is 200.

Note

The conditional expression when specified in the following statements:

- IF statement
- UNTIL phrase of PERFORM statement
- WHEN phrase of EVALUATE statement
- WHEN phrase of SEARCH statement

e. Number of split key name (Indexed file)

When specifying split key names in RECORD KEY clause or ALTERNATE RECORD KEY clause of an indexed file, the maximum number is 30. The maximum number of items specified for one split key name is 61.

f. JAPANESE phrase of INITIALIZE statement

When specifying JAPANESE in INITIALIZE statement, the maximum number of COBOL words constituting identifier of BY phrase is 47.

g. Hierarchy level of conditional compilation

Maximum hierarchy level (nested) of conditional compilation is 64.

h. Hierarchy level of EVALUATE statement and SEARCH statement

Total of the hierarchy levels for EVALUATE statement and SEARCH statement is 30.

i. Number of conditional expressions that can generate a data item

Maximum number of conditional expressions subject to conversion (m0509 or m0510) that can generate a data item within the conditional expressions in procedure division is 2,400.

j. Number of data names specified in the ASSIGN clause

The maximum number of data names specified in the ASSIGN clause is 50. If data names with the same name are specified in more than one ASSIGN clause, it counts as one.

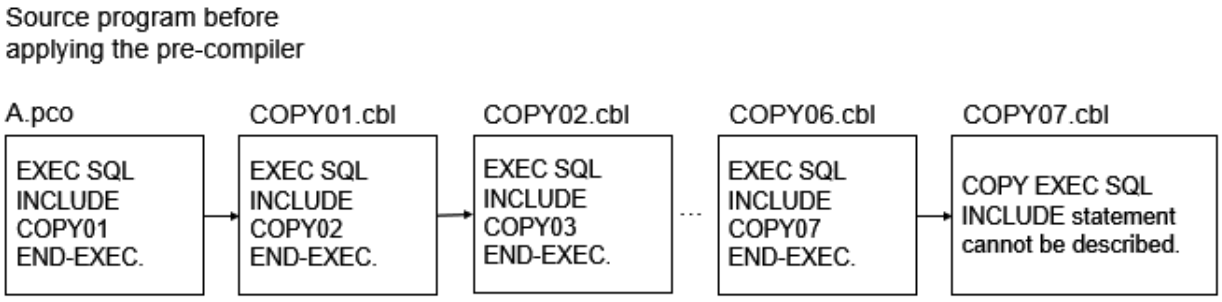
k. Number of named literal described in the report section

The maximum number of named literals described in the report section of the data division is 30,000.

l. Hierarchy level of file name control information

The maximum hierarchy level (nesting) of file name control information (#FILE information) is 8.

If the hierarchy level (nesting) of the EXEC SQL INCLUDE statement described in the source program before applying the pre-compiler exceeds 7 levels, this quantitative restriction applies.



Index