FUJITSU

Fujitsu Software NetCOBOL V13.0.0

Release Notes

Windows(64)

B1WD-3687-01ENZ0(00)
December 2025

Preface

This book explains Release Note of NetCOBOL.

Abbreviations

The following abbreviations are used in this manual:

Product Name	Abbreviation
Microsoft(R) Windows Server(R) 2025 Datacenter Microsoft(R) Windows Server(R) 2025 Standard	Windows Server 2025
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
Windows(R) 10 Home Windows(R) 10 Pro Windows(R) 10 Enterprise Windows(R) 10 Education	Windows 10
Windows(R) 8.1 Windows(R) 8.1 Pro Windows(R) 8.1 Enterprise	Windows 8.1
Red Hat(R) Enterprise Linux(R) 9(for Intel64) Red Hat(R) Enterprise Linux(R) 8(for Intel64)	Linux(64)

- In this manual, when all the following products are indicates, it is written as "Windows".
    - Windows Server 2025
    - Windows Server 2022
    - Windows Server 2019
    - Windows 11
- For 64bit Windows, use "Windows(64)."

- For 32bit Windows, use "Windows(32)."
- The COBOL development system that runs on Windows systems and develops 32bit COBOL applications is written as "NetCOBOL for Windows(32)."
- The COBOL development system that runs on Windows(64) systems and develops 64bit COBOL applications is written as "NetCOBOL for Windows(64)."
- The COBOL development system that runs on Linux(64) and develops 64bit COBOL applications is written as "NetCOBOL for Linux(64)."
- The COBOL development system that runs on Solaris systems and develops 32bit COBOL applications is written as "NetCOBOL for Solaris." Oracle Solaris on which Solaris 32bit NetCOBOL runs is referred to as "Solaris."

Third-party COBOL

- In this document, "third-party COBOL" refers to Rocket COBOL from AMC Software Japan LLC and COBOL products from the former Micro Focus.

Purpose of this book

This book explains the function addition, the trouble correction from the old edition, and information on interchangeability according to them.

It has aimed can the smoother shift of the customer who was using the old edition to this product.

Object reader in this book

An old product is introduced, and it is targeted for the shift to this product to be examined, and to be shifting.

Location of this book

Please refer to the manual of each program for a whole image or detailed information because it is fragmentary information though this book explains the function addition, the trouble correction, and information on interchangeability according to them.

Trademarks

Export Regulation

Publication Date and Edition

Publication Date and Edition	Manual Code
December 2025, Version 1.0	B1WD-3687-01ENZ0(00)

Copyright Notice

Contents

Chapter 1 Outline of Additional Functionality

New product functionality and content is outlined below by version and level.

1.1 All Products

The following Information applies to all products.

Table 1.1 Overview of Additional Functions Common to Compone

NO.	V/L	Function Name	Content	Location in Manual
1	V13.0.0	OS support	Windows Server 2025 is supported.	-
2	V12a (V12.2.0)	OS support	Windows 11 is supported.	-
3	V12a (V12.2.0)	Replacing internal components	Replaced the internal components used by the installer.	-
4	V12.2.0	OS support	Windows Server 2022 and Windows Server 2019 are supported.	-
5	V11.1.0	OS support	Windows Server 2016 is supported.	-
6	V11.0.1	OS support	Windows 10 is supported.	-
7	V11.0.0	Fujitsu Common Tools	Fujitsu Middleware Installation System and FJQSS (Information Collection Tool)	- NetCOBOL Software Release Guide - FJQSS User's Guide

Note

The V/L column lists the version and level of NetCOBOL Enterprise Edition.

For information on environments no longer supported in this version/level, refer to "2.7.3 Unsupported Environments."

1.2 NetCOBOL

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

1.2.1 Additional Functions in V13.0.0

This section describes the new features and improvements added in V13.0.0.

Integration with Other Systems

- Provides a library to help to call COBOL programs from Java.

  This allows you to work with rich features that Java supports, and utilize COBOL programs in many locations including the cloud.

  For example, you can develop a service which call a COBOL program via REST API by calling the COBOL program from a REST service written in Java.

  Refer to "COBOL Invoker for Java API Reference" and "COBOL Invoker for Java Getting Started."

  For details on accessing these documents, refer to "COBOL Invoker for Java" in "NetCOBOL Software Release Guide."

NetCOBOL Studio

- This release supports Eclipse 2024-12 (4.34). Note that, starting from V13.0.0, Eclipse is no longer bundled. Instead, a NetCOBOL Studio plugin for installation into Eclipse is provided. This allows you to use the same Eclipse environment for both COBOL and Java development.

  Refer to "NetCOBOL Studio User's Guide."

- Security is now enhanced through support for SSH port forwarding in remote debugging, enabling encrypted communication.

  Refer to "Remote Development Function " in "NetCOBOL Studio User's Guide."

- Remote development functionality now uses SSH for enhanced security. In addition to the enhanced security targeting Linux(64) introduced in V11, Solaris and Windows(64) are now also targeted for these security enhancements. Remote debugging security has also been enhanced.

  Refer to "Remote Development Function" in "NetCOBOL Studio User's Guide."

Support for the Replacement of Text by Partial Word of International Standard COBOL 2014

- The source text manipulation (COPY and REPLACE statements) can now specify partial word supported by the international standard COBOL 2014.

  Refer to "COPY Statement" and "REPLACE Statement" in "NetCOBOL Language Reference."

Enhancement to CORRESPONDING Phrase Specification in ADD and SUBTRACT Statements

- The specification for the CORRESPONDING phrase in ADD and SUBTRACT statements has been enhanced to allow group items containing data items other than numeric items.

  Refer to "CORRESPONDING Phrase" in "NetCOBOL Language Reference."

Support for Third-Party COBOL User-Defined Word Compatibility Mode

- By specifying the compiler option MF or USERWORD, third-Party COBOL specific user-defined words can be defined in NetCOBOL.

  This makes it easier to migrate from third-party COBOL.

  Refer to the following sections:

  - "NetCOBOL User's Guide":
    - "MF (third-party COBOL compatibility mode)"
    - "USERWORD (specifying user-defined word mode)"
    - "COPYHYPN (specifying characters added/replaced by the compiler in a COPY statement)"
  - "NetCOBOL Language Reference":
    - "User-defined word Compatibility Mode"

Support for Zoned Decimal Compatibility Mode in Other Formats

- Support the third-party COBOL compatible format and the 88 consortium compatible format as the format expressing the sign part of signed zoned decimal data item without the SEPARATE phrase.

  This makes it easier to migrate from third-party COBOL.

  Refer to the following sections In "NetCOBOL User's Guide":

  - "DECIMAL (internal format of zoned decimal items with no SEPARATE phrase)"
  - "DECIMALCHK (mixed checking of zoned decimal format at runtime)"
  - "Compatibility of Zoned Decimal Modes with Other Formats"

Third-Party COBOL Compatibility Features

- The TRAILING phrase of the INSPECT statement can now search for a string from the rightmost position of a character string.

  This makes it easier to migrate from third-party COBOL.

  Refer to "INSPECT statement (third-party COBOL Native)" in "NetCOBOL Language Reference."

Expanded Locations for Integer and Numeric Functions

- Integer function and numeric function can now be described as sending items of MOVE statements. You can now describe the LENGTH function wherever you can specify an integer.

  This makes it easier to migrate from third-party COBOL.

  Refer to "Function Types" and "LENGTH Function" in "NetCOBOL Language Reference."

COMMAND-LINE Support

- You can now manipulate command line data using the ACCEPT and DISPLAY statements with the function name COMMAND-LINE.

  This makes it easier to migrate from third-party COBOL.

  Refer to the following sections:

  - "NetCOBOL User's Guide"

    "Command Line Data Handling Function"

  - "NetCOBOL Language Reference"

    "Command Line Data"

Screen Functions

- ACCEPT and DISPLAY statements now support data items in the linkage section.

  This makes it easier to migrate from third-party COBOL.

  Refer to "Screen Functions" in "NetCOBOL Language Reference."

- Pressing [Shift] + [TAB] now moves to the previous field.

  Refer to "Keys Used for Screen Operations" in "NetCOBOL User's Guide."

Embedded Exception Declaration Specification Extension for Database (SQL)

- Warnings can now be detected in embedded exception declarations in the database (SQL). You can now also write a PERFORM statement in addition to the GO TO statement when an exception occurs.

  This allows processing to continue immediately after a SQL statement after a warning or error has been addressed.

  Refer to the following sections:

  - "NetCOBOL Language Reference":

    - " Embedded Exception Declaration"

  - "NetCOBOL User's Guide":

    - "SQLEXCP (specifying the notification area referenced in embedded exception declarations)"

    - "SQLWARNING (determines whether embedded exception declaration warnings are detected) "

Support for ODBC Positioning Updates

- Position updates are now available when using the Enterprise Postgres ODBC driver.

  Refer to "Creating an ODBC Information File" in "NetCOBOL User's Guide."

Specifying Behavior When Moving NULL Characters

- When accessing a database through ODBC, you can now specify what happens when a NULL character is set.

  Refer to "Creating an ODBC Information File" in "NetCOBOL User's Guide."

ACCEPT/DISPLAY Function

- You can now change the input mode of an ACCEPT statement that is associated with no function name or the function name SYSIN

  Refer to "@CBR_ACCEPT_FROM_SYSIN (Specify the input mode for ACCEPT FROM SYSIN at runtime) " in "NetCOBOL User's Guide."

- You can now suppress messages when executing the ACCEPT statement associated with the function name CONSOLE.

  Refer to "@CBR_JMP0201A_MESSAGE (Specify the suppression of ACCEPT FROM CONSOLE message at runtime)" in "NetCOBOL User's Guide."

Tab Reading in Line Sequential Files

- Tabs within records in line sequential files can now be read as part of the record data.

  Refer to "Processing Line Sequential Files" and "@CBR_FILE_TAB_READ (Specify to enable tabs handling for line sequential files)" in "NetCOBOL User's Guide."

Memory Check Function

- You can now check runtime system area before or after executing a CALL statement.

  Refer to "Using the Memory Check Function" in "NetCOBOL Debugging Guide."

Object Comparison Tool

- Improved comparison between object files and object programs linked to executable files. It is now easier to identify the source file.

  Information of compiler options and compiler fix that do not affect execution are not compared.

Improved Makefiles Created by the cobmkmf Command

- For remote development of NetCOBOL Studio for Windows, the Makefile created by the Makefile creation function can be used in the build by the make command on the server side for the COBOL project whose extension association was changed in the File Content. Refer to "cobmkmf Command" in "NetCOBOL User's Guide."

1.2.2 Additional Functions in V12.2.0

This section describes the new features and improvements added in V12.2.0.

Pre-Compilation Source Convert Function

- The pre-compilation source convert function converts the source program written in the third-party COBOL syntax, into a source program which can be compiled by NetCOBOL.

  Using this function, it can automatically convert source programs and can perform migration efficiently.

  Refer to "NetCOBOL User's Guide (Third-party COBOL Resource Migration)" and the following sections in "NetCOBOL User's Guide":

  - " -CV(Specify use of the pre-compiler source conversion function)"
  - " FLAG(diagnostic message level)"
  - " PRECONV(Determines whether to use the pre- compiler source conversion function)"
  - " COBPRECONV Command"

Third-party COBOL File Migration Tool

- Third-party COBOL file migration tool converts the COBOL file created by an application written in third-party COBOL syntax into a format accessible by NetCOBOL.

  Using this tool, you can automatically convert COBOL files and can perform migration efficiently.

  Refer to "Migration COBOL File" in "NetCOBOL User's Guide (Third-party COBOL Resource Migration)."

Program Repair Support Function

- In the program repair, you can identify the source file used to create an executable file when multiple source files may have been used during development. In build, you can skip the option to re-compile programs that are not affected by the change.

  Refer to the following sections in "NetCOBOL User's Guide":

  - "Program Repair Support Function"
  - "OBJECT(whether an object program should be output)"
  - "COBCMPO Command"

Dump Function

- Support for dump information written from the executable file and object program. This allows you to check the following information:

  - Compiler options specified when creating an object program.
  - Object programs constituting an executable file.
  - Recommended compiler options to be unified between programs.

  Refer to "COBDUMP Command" in "NetCOBOL User's Guide."

Data Item Boundary Specification Support

- Compiler can align the data item with SYNCHRONIZED clause specified on NetCOBOL native boundary or system boundary.

  Refer to "SYNC (handling of data item boundary with SYNCHRONIZED clause specified)" in "NetCOBOL User's Guide."

Cursor Operation Support at Transaction Completion

- In database (SQL) function, the cursor operation can be specified at transaction completion.

  Refer to "Creating an ODBC Information File" in "NetCOBOL User's Guide."

Pipe Support for Standard Input

- Support for reading data from pipe as standard input.

  Refer to "Command Prompt Windows" in "NetCOBOL User's Guide."

1.2.3 Additional Functions in V12.0.0

COMP-6 Support

- The data of packed decimal data item without sign half-byte of the third-party COBOL native can be used by describing USAGE IS COMPUTATIONAL-6.

  Refer to "Packed decimal data item without sign half-byte" in "NetCOBOL Language Reference."

Fujitsu Mainframe floating-Point Support

- The data of Fujitsu mainframe format floating-point format can be operated by using the Fujitsu mainframe floating-point arithmetic emulator.

  Refer to the following sections in "NetCOBOL User's Guide":

    - "FLOAT (internal format of internal floating-point item)"
    - "FLOATCHK (mixed check of internal format of internal-floating point items at execution time) "
    - "Format of floating-point"

LE Subroutine Support

- The LE subroutines of IBM can be used.

  Refer to the "LE Subroutines User's Guide."

NetCOBOL Studio

- Eclipse 4.6 workbench can be used in NetCOBOL Studio. Under this environment, the following operability is improved.
    - Split editor view
    - Enlarging or reducing font size by shortcut key
    - The icon of the tool bar is enlarged even on high resolution screen.

  This version of NetCOBOL Studio supports newer Eclipse versions. For details, refer to "Overview" in "NetCOBOL Studio User's Guide."

- Support the break using the conditional expression at debugging time. This enables efficient debugging.

  Refer to "Debugging Function" in "NetCOBOL Studio User's Guide."

Enhancement of Development Environment

- The following functions were enhanced.
    - Extensions other than .cob, .cbl, and .cobol can be treated as extensions of COBOL source files.
    - For remote development, makefile creation and remote debugging, the following files are possible.
        - The COBOL libraries with extensions other than .cbl.
        - The descriptor files with extensions other than .smd and .pmd.
    - NetCOBOL Studio can treat files stored in a subfolder under the workspace.

      For remote development, NetCOBOL can create a subdirectory with the same name as subfolder on the server side and send the following files.
        - The COBOL source files, the COBOL library files, the descriptor files stored in subfolder under the workspace.

  Refer to "File Content" in "NetCOBOL Studio User's Guide."

Switch of Compiler Diagnostic Message Format

- Compiler diagnostic message format can be switched. The message is displayed in the standard error output.

  Refer to "Setting Environment Variables" in "NetCOBOL User's Guide."

Support for the Relational Operator <>

- This release adds support for the relational operator <>.

  Refer to "Relation Condition" in "NetCOBOL Language Reference."

1.2.4 Additional Functions in V11.1.0

Enhanced file-Identifier and File-Identifier Literal

- Up to 30 Characters can now be specified for file-identifier.

  The file-identifier literal can be treated as an environment variable name.

Refer to "FILELIT (file-identifier literal handling)" in "NetCOBOL User's Guide" and "ASSIGN Clause (Sequential File, Relative File, and Indexed File)" in "NetCOBOL Language Reference."

Third-party COBOL Synonym Support

- The third-party COBOL native synonyms can be used in NetCOBOL by specifying the compiler option MF.

  Refer to "MF (third-party COBOL compatibility mode)" in "NetCOBOL User's Guide" and "Synonym Compatibility Mode" in "NetCOBOL Language Reference."

PRINTER_n Support

- In the ASSIGN clause, PRINTER_n phrase is supported. n for PRINTER_n is an integer from 1 through 99.

  Refer to "ASSIGN Clause (Sequential File, Relative File, and Indexed File)" and "The WRITE Statement (Object-Oriented Programming)" in "NetCOBOL Language Reference."

Enhanced cobmkmf Command

- The extensions other than default (*.cob, *.cobol) can be used as an extension of the source files that the cobmkmf command handles.

  Moreover, the files in the subdirectory can be included in the build target.

  Refer to "cobmkmf Command" in "NetCOBOL User's Guide."

Supports Specification of Data Area Allocation Method

- Specify whether the memory which allocates the program data area is acquired statically or dynamically.

  Refer to "Program data area allocation" and "DATAAREA(Specifies the method of data area allocation)" in "NetCOBOL User's Guide."

1.2.5 Additional Functions in V11.0.1

Eclipse 4.3 Workbench Support

- Eclipse 4.3 workbench can now be used in NetCOBOL Studio, in which view layout can be changed and recent plugin can be used.

  This version of NetCOBOL Studio supports newer Eclipse versions. For details, refer to "Overview" in "NetCOBOL Studio User's Guide."

Free Format COBOL Editor Support

- Free format style COBOL editor can now be used in NetCOBOL Studio.

  Refer to "Reference Formats" and "SRF compile option" in "NetCOBOL Studio User's Guide."

Security Enhancement in Remote Development

- SSH Port Forwarding enhances the security in remote developing on Linux(64).

  This version of NetCOBOL Studio expands the targets of security enhancement.

  For details, refer to "Remote Development Function" in "NetCOBOL Studio User's Guide."

1.2.6 Additional Functions in V11.0.0

UTF-32 Support

- Encoding form UTF-32 can now be used.

  Refer to "Unicode" in "NetCOBOL User's Guide."

COBOL Resource Project Function Support

- A "COBOL resource project" is used for management of the library file and the descriptor file on the NetCOBOL Studio.

  Refer to "COBOL Resource Project" in "NetCOBOL Studio User's Guide."

COBOL Solution Project Function Support

- A "COBOL solution project" is used for management of multiple projects on the NetCOBOL Studio.

  Refer to "COBOL Solution Project" in "NetCOBOL Studio User's Guide."

Project configuration conversion command

- Provide the project configuration conversion command as a transfer support tool in order to convert the project in Project Manager to the project in NetCOBOL Studio.

  Refer to "Transition of Project According to Project Configuration Conversion Command" in "NetCOBOL Studio User's Guide."

Default Values for Paper Size and Print Format Specifications

- Paper size and printing format can now be specified in print information file.

  Refer to "Print Information File" in "NetCOBOL User's Guide."

1.3 J Adapter Class Generator

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

NOTE: The version in parentheses shows the version level in the NetCOBOL series.

1.3.1 Additional Functions in V12.1.0 (V12.2.0)

Explicit Specification of JavaVM Path

- New function to specify JavaVM (jvm.dll) path explicitly. This function can be used in the new Java environment.

  Refer to the following sections in "J Adapter Class Generator User's Guide":

  - "-vm JavaVM-path" of "Options"
  - "Environment variable COBJNI_JAVA_VM" of "JVM-INIT Method (factory method)"

1.3.2 Additional Functions in V11.0.0 (V11.0.0)

Unicode Character for File Path Character String

- The Unicode character can be used for the file path character string specified for a command line argument and an optional file.

1.4 PowerFORM

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

1.4.1 Additional Functions in V12.2.0

Enhanced Print Features

- New era (Reiwa) is supported.

1.4.2 Additional Functions in V12.0.0

Enhanced Print Features

- The era name of Japanese calendar can be customized.

  Refer to "GENGO (Japanese era name)" in "PowerFORM Runtime Reference."

1.4.3 Additional Functions in V11.1.0

Enhanced Print Features

- Printer paper can now be specified when the specified paper was not supported with the output printer.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

1.4.4 Additional Functions in V11.0.0

Unicode(UTF-32) Support

- UTF-32 data can now be used in COBOL applications.

  Refer to "How to Use PowerFORM RTS" in "PowerFORM Runtime Reference."

Encoding Systems for Print Information File

- Print Information File in UTF-8 with BOM can now be used in user created Unicode COBOL applications.

  Refer to "How to Use PowerFORM RTS", "Printer Information File", "Troubleshooting and Error Recovery" in "PowerFORM Runtime Reference."

31-digit Support

- Up to 31 digits can now be used for numeric fields in COBOL applications.

  Refer to "Output Field Decorations" in "PowerFORM Runtime Reference."

Form Exporting

- The Form can now be outputted to PDF. And The character (surrogate pair) added by character-code standard "JIS X 0213:2004(JIS2004)" can be output.

  Refer to "Extended Functions" and "Printer Information File" in "PowerFORM Runtime Reference."

Enhanced Print Features

- Print options are provided to define the behavior when output printer is omitted and when it fails in opening the specified output printer.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- Print options are provided to define the behavior when the specified paper was not supported with the printer device.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- Print options are provided to define the behavior when the specified paper feed was not supported with the printer device.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- The processing time at expansion/reduction print and the print preview can be shortened.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- The processing time of the Form output using the fixed-width font can be shortened.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- When the user character-code is UNICODE, the table of full-width/half-width character used by outputting the UNICODE character can be specified, and customize at the character-code level can be specified.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

- The Form Descriptor (free frame) can be used.

  Refer to "Extended Functions" in "PowerFORM Runtime Reference."

- The new function added in PowerFORM V11 can be used.

  Refer to "What's New in PowerFORM V11" in "PowerFORM Getting Started."

Support Locale

- The following 4 locales outputs are supported.

  - Japanese
  - English
  - Chinese (simplified Chinese)
  - Portuguese

  Refer to "4.4.4 PowerFORM Runtime."

UTC (Coordinated Universal Time) support

- UTC (Coordinated Universal Time) can be specified.

  Refer to "Printer Information File" in "PowerFORM Runtime Reference."

1.5 Fujitsu Mainframe Floating-Point Arithmetic Emulator

The Fujitsu mainframe floating-point arithmetic emulator is provided from V12 or later.

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows

1.5.1 Additional Functions in V12.0.0

Fujitsu Mainframe Format Floating-Point Support

- The data of Fujitsu mainframe format floating-point format can be operated by using the Fujitsu mainframe floating-point arithmetic emulator.

  Refer to the following sections in "NetCOBOL User's Guide":

  - "FLOAT (internal format of internal floating-point item)"
  - "FLOATCHK (mixed check of internal format of internal-floating point items at execution time)"
  - "Format of floating-point"

1.6 PowerBSORT

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

NOTE: The version in parentheses shows the version level in the NetCOBOL series.

1.6.1 Additional Functions in V9.0.0 (V13.0.0)

PowerBSORT development has been migrated to a new development platform.

For information about environments no longer supported in this version, refer to "Features No Longer Provided in V9.0.0:."

1.6.2 Additional Functions in V8.0.1 (V12.2.0)

Data format

- The Fujitsu mainframe format floating-point was supported with NetCOBOL.

  Refer to "Data format" and "Data forms that can be specified in each field" in "PowerBSORT User's Guide."

1.6.3 Additional Functions in V8.0.0 (V12.0.0)

Data format

- The COMP-6 format was supported with NetCOBOL.

  Refer to "Data format" and "Data forms that can be specified in each field" in "PowerBSORT User's Guide."

Chapter 2 Information on Interchangeability

Here, information on the interchangeability changed from the version and level before is described.

2.1 NetCOBOL Development Environment

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows

2.1.1 NetCOBOL Studio Configuration Changes

Content

In NetCOBOL V12.2.0 or earlier, NetCOBOL Studio was provided with a bundled Eclipse platform. From V13.0.0 or later, the Eclipse platform is no longer bundled. Instead, a NetCOBOL Studio plugin is provided for installation into a separately obtained and installed Eclipse IDE package. The NetCOBOL Studio launch menu item has been removed from the Start menu.

The following differences exist between NetCOBOL Studio (V12.2.0 or earlier) and an Eclipse IDE with the NetCOBOL Studio plugin installed:

NetCOBOL Studio (V12.2.0 or earlier):

- Launched by clicking the NetCOBOL Studio icon in the Start menu.
- Workspace and launch options could be specified on the startup screen.
- Default perspective: COBOL.

Eclipse IDE (V13.0.0 or later):

- Launched by executing eclipse.exe in the installed Eclipse IDE folder.
- Workspace can be specified in the startup screen (Eclipse IDE Launcher). Launch options must be specified when executing eclipse.exe.
- Default perspective: Java.

Action

Install Eclipse and the NetCOBOL Studio plugin by referring to the following:

- "Eclipse IDE for Use with NetCOBOL Studio" in "NetCOBOL Software Release Guide"
- "Installing Eclipse IDE and NetCOBOL Studio Plugin" in "NetCOBOL Installation Guide"

To use NetCOBOL Studio functionality, launch the Eclipse IDE by executing eclipse.exe in the Eclipse folder where the NetCOBOL Studio plugin is installed. Specify Eclipse launch options in the eclipse.exe command.

When using the NetCOBOL for Windows(64), specify the -cobolarch x64 option when starting Eclipse.

Example:

```
<Eclipse IDE installation folder>\eclipse.exe -cobolarch x64
```

The default perspective in the Eclipse IDE is Java. To open the COBOL perspective, select Window > Perspective > Open Perspective > Other, choose COBOL in the "Open Perspective" dialog, and click OK.

COBOL development functionality within the COBOL perspective remains the same as in V12.2.0.

Workspaces created in V12.2.0 or earlier can be upgraded by opening them in the Eclipse IDE with the NetCOBOL Studio plugin installed. For details, refer to "Handling of Workspace and Project" in "NetCOBOL Studio User's Guide."

2.1.2 Removal of the Navigator View in NetCOBOL Studio

Content

The Navigator view, provided in NetCOBOL Studio in NetCOBOL V12.2.0 or earlier, has been removed in V13.0.0.

The Navigator view, provided by the Eclipse platform underlying the NetCOBOL Studio plugin, displayed the resources within a workspace in a hierarchical structure.

As noted in "1.2.1 Additional Functions in V13.0.0," V13.0.0 supports Eclipse 2024-12 (4.34). Because the Navigator view was removed from Eclipse 2024-12 (4.34), it has also been removed from NetCOBOL Studio in V13.0.0.

Action

Use the Project Explorer view instead of the Navigator view.

For details, refer to "Project Explorer View" in "NetCOBOL Studio User's Guide."

2.1.3 Removal of CVS (Concurrent Versions System) File Sharing Functionality in NetCOBOL Studio

Content

The CVS (Concurrent Versions System) file sharing functionality provided in NetCOBOL Studio in NetCOBOL V12.2.0 or earlier is no longer available in V13.0.0.

As noted in "1.2.1 Additional Functions in V13.0.0," V13.0.0 supports Eclipse 2024-12 (4.34). Because CVS-related functionality was removed from Eclipse 2024-12 (4.34), it is also no longer available in NetCOBOL Studio in V13.0.0.

Action

Consider using a file sharing mechanism other than CVS. Refer to "File Sharing with Git Repositories" in "NetCOBOL Studio User's Guide."

2.1.4 Change in Default Text File Encoding Setting in NetCOBOL Studio Workspaces

Content

The default text file encoding setting in NetCOBOL Studio workspaces has changed. In NetCOBOL Studio based on Eclipse 4.6 (V12.2.0 or earlier), the default setting was "Cp1252" if the system locale of the operating system was set to English. In NetCOBOL Studio based on Eclipse 2024-12 (4.34) (V13.0.0 or later), the default setting is now "UTF-8."

Consequently, as described in "Wizards" in "NetCOBOL Studio User's Guide", the default text file encoding specified in the COBOL Project Generation Wizard has also changed from "ACP(Cp1252)" in V12.2.0 or earlier to "UTF-8" in V13.0.0 or later.

Information

- As described in "Setting compile options" in "NetCOBOL Studio User's Guide," the compiler option SCS, which specifies the character encoding for COBOL source files within a COBOL project, is automatically determined from the text file encoding setting.
- As described in "Importing Project" in "NetCOBOL Studio User's Guide", if "Inherited from container (I) (<encoding setting>)" is set for the text file encoding of a COBOL project, the workspace text file encoding setting is applied.

Action

To maintain compatibility with NetCOBOL Studio V12.2.0 or earlier, change the text file encoding setting to "windows-1252"(*) after starting the workbench. Refer to "Setting the Text File Encoding for the Workspace" in "NetCOBOL Studio User's Guide."

*: The display of the text file encoding setting value for the workspace has changed from "Cp1252" in Eclipse 4.6 NetCOBOL Studio V12.2.0 or earlier) to " windows-1252" in Eclipse 4.34 (V13.0.0 or later).

2.1.5 Incompatibilities Due to Security Enhancements for the Remote Development Function

From NetCOBOL V13.0.0 or later, an SSH server (hereinafter referred to as sshd) is used for remote build connections.

2.1.5.1 Changes to Required Services on the Server Side

Content

The following services, which were previously required on the server side, are no longer necessary:

- Servers on which NetCOBOL for Windows(64) or NetCOBOL for Linux(64) is installed:
    - NetCOBOL Remote Development Service
- Servers on which NetCOBOL for Solaris is installed:
    - ftpd/rexec service

Accordingly, the GUI of the server information settings dialog in NetCOBOL Studio has also been changed. The default fingerprint algorithm for new server connections has been changed from md5 to sha256.

Actions:

Connecting to a server using the NetCOBOL Remote Development Service or the ftpd/rexec service is no longer supported. Use sshd for the server-side service. Also, server information configured in NetCOBOL Studio V12.2.0 or earlier may not be usable. If the error "Cannot connect to the SSH Service. Verify that the SSH Service is running and make sure you have permission to connect to the SSH port." appears when connecting to the server, you must reconfigure the server information. For details, refer to "Remote Development Function" in "NetCOBOL Studio User's Guide."

2.1.5.2 Discontinuation of SSH Port Forwarding Connection

Content

The SSH port forwarding connection, which was available for servers with NetCOBOL for Linux(64) installed, has been discontinued and integrated into the SSH connection. Existing SSH port forwarding settings are carried over to the SSH connection settings.

Action:

No action is required.

2.1.6 Change the File Association by Using PRINTER_n Phrase

Content

- V11.0 or earlier

  The ASSIGN clause with PRINTER_n phrase associates file-name with a physical file.

- V11.1 or later

  The ASSIGN clause with PRINTER_n phrase associates file-name with a printing device.

Action

Change PRINTER_n to another name.

2.1.7 Change Default Link Option /SUBSYSTEM

Content

The default of link option /SUBSYSTEM was changed.

Impact

When the following functions are used, the output character string might not be displayed in the window with this link option.

- COBOL Console Window
- Screen Function

Action

If you use COBOL console window or screen function, please specify following LINK option when you link main programs.

- When compiler option MAIN(WINMAIN) is specified for the main program, it is WinMain type.

  When compiler option MAIN(MAIN) is specified for the main program, it is main type.

  [WinMain type]

  /SUBSYSTEM:WINDOWS and 5.02

  [main type]

  /SUBSYSTEM:CONSOLE and 5.02

- If you use screen function it is possible to display it correctly by specifying environment variable information @ScrnSize and changing size of logical screen.

2.1.8 Reference Format Setting in NetCOBOL Studio

Content

In NetCOBOL Studio (Eclipse 4.3 based) V11.0.1 and NetCOBOL Studio V11.1.0 or later, the reference format settings and tab width settings of the editor are no longer automatically reflected in the compiler option SRF and TAB of the project, respectively.

Action

To restore the consistency to NetCOBOL Studio V11.0.1 or earlier (the Eclipse 3.4 workbench), select "SRF and TAB compile option setting to be consistent with the applicable editor setting" checkbox.

2.1.9 Runtime Code-Set When Compiler Option ENCODE is Specified

Content

Added the compiler option ENCODE to define the encoding form of the data item.

With the specification of compiler option ENCODE, the runtime code set is changed.

- When the compiler option RCS is explicitly specified.
  - The runtime code-set becomes the code-set that is specified with the compiler option RCS.
- When the compiler option RCS is not specified explicitly.
  - When the compiler option ENCODE is specified explicitly, then the runtime code-set becomes the Unicode.
  - When the compiler option ENCODE is not specified explicitly, then the runtime code-set becomes the ANSI code page.

Conditions

1. In compiler versions V10.5.0 or earlier, the compiler option RCS (SJIS) is specified explicitly or implicitly and it is a program asset.
2. And, in compiler versions V11.0.0 or later, without specifying the compiler option RCS (SJIS) explicitly, for the program assets of 1 above, the compiler option ENCODE(SJIS[,SJIS]) is described explicitly and recompiled.

Impact

The impact corresponding to the above conditions is as follows.

- An error (JMP0081I-U) occurs in the following situations.
    - When the recompiled program is called from a program that is not recompiled.
    - When a program that is not recompiled is called from a program that is recompiled.
- During execution after the main program is recompiled, the source becomes the Unicode.

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

Action

Create the target program without explicitly specifying the compiler option ENCODE.

Moreover, in regards to the repository, keep the compiler option RCS and the compiler option ENCODE specified on the reference and remote side the same.

2.1.10 Location of SQLCODE/SQLMSG/SQLERRD Definition

SQLCODE, SQLMSG, and SQLERRD can now be defined outside of the SQL declaration section. Under the conditions listed below, SQL statement runtime information was not stored in V10.1.0. In V10.2.0 and later, runtime information is stored in SQLCODE, SQLERRD, and SQLMSG.

1. SQLSTATE is described inside the SQL declaration section, and
2. SQLCODE, SQLERRD, or SQLMSG are described outside the SQL declaration section, and
3. SQL statements are executed, and
4. SQLCODE, SQLERRD, or SQLMSG are referred to in the PROCEDURE DIVISION, and
5. Conditions 1 through 4 are occurred in the same compilation unit.

In V10.2.0 and later, SQLCODE, SQLERRD, and SQLMSG are reserved names used to fetch SQL statement runtime information. If you have variables by those names that are used for any other purpose, the results are unpredictable. In this case, change the names of those variables.

2.1.11 Result of Addition and Subtraction of Zoned Decimal Item Outside Regulations

Under the following conditions, the execution result of V10.1.0 and that of V10.2.0 and later are different.

1. There is a binary operation of addition or the subtraction.
2. The number of identifier of the result is 1.
3. [NOT] ON SIZE ERROR is not specified.
4. ROUNDED is not specified.
5. Two operands are zoned decimal and numbers of digits are from 16 to 18.
6. The accuracy of intermediate result (*) of binary operation is from 17 to 19 digits.
7. Either of representation of two operands is wrong as zoned decimal.

    *: Refer to the "NetCOBOL Language Reference" about the accuracy of intermediate result.

Example

```
           WORKING-STORAGE SECTION.
           01 DATA-ZONE1    PIC S9(18).
           01 DATA-ZONE2    PIC S9(18).
           01 DATA-2 REDEFINES  DATA-ZONE2 PIC X(18).
           01 DATA-ZONE3    PIC S9(18).
           01 DATA-3 REDEFINES  DATA-ZONE3 PIC X(18).
           PROCEDURE DIVISION.
               MOVE X"FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF" TO DATA-2  DATA-3  .....(a)
               COMPUTE DATA-ZONE1 = DATA-ZONE2 + DATA-ZONE3
               DISPLAY DATA-ZONE1
```

In the above program, save area of alphanumeric item is allocated in zoned decimal. When (a) is executed, wrong representation (0xFFFF...FFFF) is set to DATA-ZONE2 and DATA-ZONE3 as zoned decimal.

- V10.1.0

```
    +333333333333333330
```

- V10.2.0 and later

```
    +545555555455555554
```

The result of the arithmetic statement including the operand to which wrong internal format is set as zoned decimal is not provided for. Please correct the part where wrong internal format is set.

The outline of the representation that zoned decimal is correct is as follows.

Data item	Zone part	Numeric part	Sign part
Zoned decimal (SEPARATE)	3	0 to 9	2B,2D
Zoned decimal (no SEPARATE)	3		4,5

2.1.12 Interchangeable Information Regarding Bug Fixes

This section describes changes in NetCOBOL development environment behavior caused by defect fixes implemented in NetCOBOL V10 or later. V/L indicates the range of versions where the defect existed.

PH23981

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, one of the following issues could occur. This issue has been fixed.

- The COBOL program compiled without errors when a compilation error should have occurred.
- A runtime error or incorrect result occurred during the execution of the COBOL application.

[Conditions]

1. One of the following is in effect for the compiler option ENCODE (*1):
   - ENCODE(UTF8)
   - ENCODE(UTF8,UTF16)
   - ENCODE(UTF8,UTF32)
   - ENCODE(UTF8,UTF16,LE)
   - ENCODE(UTF8,UTF16,BE)
   - ENCODE(UTF8,UTF32,LE)
   - ENCODE(UTF8,UTF32,BE)
2. A alphabet-name corresponding to one of the following is declared in the ALPHABET clause:
   - UTF16
   - UTF16LE
   - UTF16BE
   - UTF32

    - UTF32LE
    - UTF32BE

3. The alphabet-name from condition 2 is specified in the ENCODING clause.
4. One of the following file declarations ([A] to [D]) is made:

[A]

- A-1. A print file without FORMAT clause or a line sequential file is declared.
- A-2 A record description entry associated with the file in A-1 contains a data description entry with an ENCODING clause that satisfies condition 3.
- A-3. The data description entry in A-2 is a national item or contains a national item.
- A-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in A-1.

[B]

- B-1. A line sequential file is declared.
- B-2. The record description entry associated with the file in B-1 contains only one elementary item.
- B-3. The data description entry in B-2 has an ENCODING clause that satisfies condition 3.
- B-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in B-1.

[C]

- C-1. A print file with FORMAT clause, or a presentation file where a display or printer is specified in the destination type, is declared.
- C-2. A record description entry associated with the file in C-1 contains a data description entry with an ENCODING clause that satisfies condition 3 (*2).
- C-3. The data description entry in C-2 is a national item or contains a national item.
- C-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in C-1.

[D]

- D-1. A line sequential file or a presentation file where display or printer is specified in the destination type is declared.
- D-2. A WRITE statement with a FROM phrase is coded for the file in D-1.
- D-3. The data item specified in the FROM phrase of the WRITE statement in D-2 is a national item or contains a national item.
- D-4. An ENCODING clause that satisfies condition 3 is specified in either of the following:
    - (a) The data description entry specified in the FROM phrase of the WRITE statement in D-2 (*2)
    - (b) The file description entry of the file in D-1
- D-5. The encoding form in (a) and (b) are different.

*1: The compiler option ENCODE shown in condition 1 is in effect when the compiler option is specified, or when the compiler option RCS(UTF16) or RCS(UCS2) is specified.

*2: Includes cases where the ENCODING clause is specified in a COPY statement with IN/OF XMDLIB.

[Impact]

From V13.0.0 or later, the following messages will be output during compilation for programs where records associated with files and the FROM phrase of WRITE statements contain multiple national encoding forms:

- JMN5789I-S There are multiple encodings that are mixed in the record display file definition, which is specified in printer device or display device in the destination type and printing file. Only one type of encoding can be specified in each alphanumeric data item and national item.

- JMN5791I-S It is not possible to include multiple encodings at the same time in the records of the Line file.
- JMN3565I-S When compilation option RCS(UTF16) or ENCODE is specified, @1@ cannot be specified in a WRITE statement for a @2@. @1@ is different from the encoding of the @2@.

  @1@: identifier

  @2@: print file, line sequential file, presentation file

[Action]

To avoid this issue, ensure that records associated with files and the FROM phrase of WRITE statements contain only one national encoding form.

PH23738

- V/L: V10.1.0 to V12.2.0A

When any of the following conditions ([Condition A] to [Condition C]) are met, one of the following symptoms may occur. This issue has been fixed.

[Symptoms]

- At compile time, incorrect messages (JMN2112I-S, JMN2113I-S, JMN2245I-S, JMN2247I-S, JMN2832I-S, JMN5400I-U) are output and the object program is not generated.
- At compile time, an incorrect message (JMN2343I-W) is output.
- At compile time, a necessary message (JMN2113I-S) is not output and the object program is generated.
- At runtime, a file with an incorrect format is generated.
- At runtime, a file open error (JMP0310I-I/U) occurs.
- At runtime, incorrect results are obtained when reading a file.
- At runtime, an error (JMP0019I-U, JMP0022I-U) occurs for an EXTERNAL file or data, and the program terminates abnormally.

[Condition A]

A record description entry is coded that meets the following conditions:

1. It has a subordinate group item.
2. The subordinate group item has a subordinate item with a TYPE clause.
3. The group item in condition 2 is redefined.

Example:

```
01 A.
   02 A-1.                 *> Corresponds to condition 2.
      03 A-1-1 TYPE T-1.
   02 A-2 REDEFINES A-1. *> Corresponds to condition 3.
      03 A-2-1 TYPE T-2.
      03 A-2-2 PIC X(8).
```

[Condition B]

A file description entry or a sort-merge file description entry is coded that meets the following conditions:

1. The record description entry associated with the file has a subordinate item with a TYPE clause.
2. The record description entry in condition 1. also meets one of the following conditions:
   - It is also has a subordinate item without a TYPE clause.
   - It meets the conditions of [Condition A].

Example:

```
FD B1.
01 B1-1.
   02 B1-1-1 TYPE T-1.
   02 B1-1-2 PIC X(8).
FD B2.
01 B2-2.     > Corresponds to condition 1 and condition 2(Condition A).
   02 B2-2-1.
      03 B2-2-1-1 TYPE T-1.
   02 B2-2-2 REDEFINES B2-2-1.
      03 B2-2-2-1 TYPE T-2.
      03 B2-2-2-2 TYPE T-3.
```

[Condition C]

A file description entry or a sort-merge file description entry is coded that meets the following conditions:

1. Two or more record description entries are coded for the file.
2. At least one of the record description entries in condition 1. meets both of the following conditions:
   - It does not have any subordinate elementary items without a TYPE clause.
   - It does not meet the conditions of [Condition A].
3. At least one of the record description entries in 1. meets one of the following conditions:
   - It is an elementary item without a TYPE clause.
   - It is a group item that does not have any subordinate elementary items with a TYPE clause.

Example:

```
FD C.
01 C-1.                    *> Corresponds to condition 2.
   02 C-1-1 TYPE T-1.
   02 C-1-2.
      03 C-1-2-1 TYPE T-2.
      03 C-1-2-2 TYPE T-2.
01 C-2.
   02 C-2-1 PIC X(8).
   02 C-2-2 TYPE T-3.
01 C-3 PIC X(8).         *> Corresponds to condition 3.
01 C-4.                    *> Corresponds to condition 3.
   02 C-4-1 PIC X(8).
   02 C-4-2 PIC X(8).
```

PH22365

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, incorrect results occurred during the execution of a COBOL program converted using the pre-compilation source conversion function. This issue has been fixed.

[Conditions]

1. The pre-compilation source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
2. An indexed file with two or more split keys defined is declared.
3. A READ or START statement is coded for the file in condition 2.
4. A split key defined second or later is specified in the KEY phrase of the statement in condition 3.

PH17766

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function performed an incorrect conversion, resulting in incorrect execution results in the converted COBOL program.

From V13.0.0 or later, the conversion is interrupted when these conditions are met.

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:
    - (1-1) Compiling with the -CVm option specified for the COBOL command.
    - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - (1-3) Converting using the COBPRECONV command.
2. Expansion of library text included by COPY statements into the post-conversion source program is enabled by one of the following:
    - Converting using method (1-1) or (1-2).
    - Converting using method (1-3) with the EXPAND-COPY conversion option set to YES.
3. The pre-conversion source program or library file contains the following description (*):
    - (3-1) A COPY statement with a REPLACING phrase is coded.
    - (3-2) A COPY statement is coded within the library text included by the COPY statement in (3-1).

*: The description that satisfies condition 3 is specific to third-party COBOL compilers. When compiling with NetCOBOL without using the pre-compiler source conversion function, the following compiler message is output:

```
JMN1074I-S The COPY statement cannot be specified in library text referenced by a COPY statement with
a REPLACING or JOINING phrase.
```

[Impact]

From V13.0.0 or later, the following message is output during conversion for programs that meet these conditions:

- When converting using method (1-1) or (1-2):

```
JMN0503I-S PRCV-ER223S The COPY statement cannot be specified in library text referenced by a COPY
statement with a REPLACING phrase. Conversion terminated.
```

- When converting using method (1-3):

```
PRCV-ER223S The COPY statement cannot be specified in library text referenced by a COPY statement
with a REPLACING phrase. Conversion terminated.
```

[Action]

Correct the source program as indicated by the conversion message.

PH17188

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function performed an incorrect conversion, resulting in incorrect execution results in the converted COBOL program. This issue has been fixed.

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.

2. Either condition [X] or condition [Y] is met.

   [X]

   - (X1) A program containing an internal program exists.
   - (X2) A file description entry with the GLOBAL clause is declared in the program in (X1).
   - (X3) The level-01 data description entry following the file description entry in (X2) does not have the GLOBAL clause.
   - (X4) A program directly or indirectly contained in the program in (X1) contains one of the following descriptions:
     - (X4-a) The same name as the data item in (X3), or one of its subordinate data items, is specified as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause.
     - (X4-b) The data item in (X3), or one of its subordinate data items, is specified in one of the following:

       (X4-b-1) A data-name in the DYNAMIC phrase of an ASSIGN clause.

       (X4-b-2) An operand of a comparison or a move operation targeted by the pre-compiler source conversion function (*1).

   [Y]

   - (Y1) A program [A] containing two or more internal programs exists.
   - (Y2) A data item with the GLOBAL clause exists in program [B], which is directly or indirectly contained in program [A].
   - (Y3) A program [C] exists after program [B] and is indirectly contained in program [A].
   - (Y4) Program [C] is not contained in program [B].
   - (Y5) The hierarchy level (*2) of program [C] is greater than that of program [B].
   - (Y6) One of the following conditions is met:
     - (Y6-a) In program [C], the same name as the data item in (Y2), or one of its subordinate data items, is specified as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause.
     - (Y6-b) A data item with the same name as the data item in (Y2), or one of its subordinate data items, exists and meets the following conditions:

       (Y6-b-1) It is declared with the GLOBAL clause in program [A], or a program directly or indirectly containing program [A].

       (Y6-b-2) In program [C], it is specified in one of the following locations:

       (Y6-b-2-1) A data-name in the DYNAMIC phrase of an ASSIGN clause.

       (Y6-b-2-2) An operand of a comparison or a move operation targeted by the pre-compiler source conversion function (*1).

*1: The pre-compiler source conversion function targets comparisons and move operations specified in the following locations:

- A conditional expression in one of the following:
  - An IF statement.
  - The selection subject of an EVALUATE statement.
  - The UNTIL phrase of a PERFORM statement.
  - The WHEN phrase of a SEARCH statement.
- A MOVE statement.

*2: The hierarchy level of a program is a number counted as follows: the outermost program is 1; a program directly contained in the outermost program is 2; a program directly contained in a level-2 program is 3; and so on.

```
Hierarchy level: 1
+------------------------------------
| Program [A]
| 01 DATA01 PIC X IS GLOBAL.
|Hierarchy level: 2
| +----------------------------------
| | Program [B]
| |  01 DATA01 PIC 9 COMP-5 IS GLOBAL.
| +----------------------------------
|Hierarchy level: 2
| +----------------------------------
| |Hierarchy level: 3
| | +--------------------------------
| | | Program [C]
| | |
| | |  IF DATA01 = "A"
| | |  THEN
| | |
```

PH17159

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function performed an incorrect conversion, resulting in an incorrect execution results in the converted COBOL program. The issue has been fixed.

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
2. One of the following two-word sequences (*1) is coded across multiple lines:
   - (2a) "ALL" of an ALL literal and a nonnumeric literal.
   - (2b) A level number and a data-name.
   - (2c) "COPY" and a text-name or text-name-literal in a COPY statement.
   - (2d) "PIC", "PICTURE", or "IS" and the PICTURE character-string.
   - (2e) "IDENTIFICATION" and "DIVISION" in the Identification Division header, or "END" and "PROGRAM" in the end program header.
   - (2f) "CONFIGURATION" and "SECTION" in the configuration section header, or "INPUT-OUTPUT" and "SECTION" in the input-output section header.
   - (2g) "FILE" and "SECTION" in the file section header.
   - (2h) "PROCEDURE" and "DIVISION" in the Procedure Division header.
   - (2i) The name of a split-key (*2) and "=" in an indexed file.
3. The character immediately following the first word in condition 2 is one of the following separators:
   - A space.
   - A comma.
   - A semicolon.

*1: The following example shows a two-word sequence coded across multiple lines:

```
000010 IDENDIFICATION
000020 DIVISION.
000030 PROGRAM-ID.  A.
```

*2: In third-party COBOL, a name called a split key can be declared for the sequence of key items in the RECORD KEY or ALTERNATE RECORD KEY clause of an indexed file. The pre-compiler source conversion function converts descriptions of these split-keys.

PH17156

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function performed an incorrect conversion, resulting in an incorrect execution results in the converted COBOL program. This issue has been fixed.

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) compiler option specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. One of the following conditions (a. through d.) is met:
    a. An EVALUATE statement is coded that meets the following conditions:
        - (a1) One of the following selection subjects is specified in a WHEN phrase of the EVALUATE statement:

          (a1-1) Two or more selection subjects are specified using the ALSO phrase.

          (a1-2) A THRU or THROUGH phrase is specified.
        - (a2) At least one selection subject in (a1) contains a combination of operands that have the same attributes as those targeted for conversion in a comparison (*).

    b. A PERFORM statement is coded that meets the following conditions:
        - (b1) A VARYING phrase is specified.
        - (b2) One or more AFTER phrases are specified after the VARYING phrase.
        - (b3) The conditional expression in the UNTIL phrase of (b1) or (b2) is targeted for conversion in a comparison (*).

    c. The following conditions are met:
        - (c1) A PERFORM statement with a VARYING phrase is coded.
        - (c2) One or more AFTER phrases are specified in the PERFORM statement in (c1).
        - (c3) A conditional expression is coded after the PERFORM statement in (c1) in one of the following:

          An IF statement.

          The selection subject of an EVALUATE statement.

          The UNTIL phrase of a PERFORM statement.

          The WHEN phrase of a SEARCH statement.
        - (c4) The conditional expression in (c3) is targeted for conversion in a comparison (*).

    d. A PERFORM statement is coded that meets the following conditions:
        - (d1) It is a PERFORM statement with a procedure-name (an out-of-line PERFORM statement).
        - (d2) An UNTIL phrase is specified in the PERFORM statement in (d1).
        - (d3) The conditional expression in the UNTIL phrase in (d2) is targeted for conversion in a comparison (*).
        - (d4) The PERFORM statement in (d1) ends with a period separator.

*: The pre-compiler source conversion function targets conditional expressions containing a combination of operands (including combinations with the left and right operands reversed) that meet one of the following conditions:

- (A) A national item and an alphanumeric item or a zoned decimal item.
- (B) A national item and a nonnumeric literal.
- (C) A national item and the figurative constant ZERO or QUOTE.

- (D) A binary item or a packed decimal item and an alphanumeric item.
- (E) A binary item or a packed decimal item and a nonnumeric literal.

If the operands in (D) or (E) are specified in the UNTIL phrase of a PERFORM statement or the WHEN phrase of a SEARCH statement, a conversion message is output without performing conversion. Therefore, the conversion message that should be output due to this issue might not be output.

PH17155

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function incorrectly converted descriptions within the pseudo-text of COPY and REPLACE statements. This issue has been fixed.

[Conditions]

Either condition X or condition Y is met.

[X]

1. The pre-compiler source conversion function is used by one of the following methods:
    - (1-1) Compiling with the -CVm option specified for the COBOL command.
    - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - (1-3) Converting using the COBPRECONV command.
2. A REPLACE statement is coded.
3. One of the following, which are targeted for conversion by the pre-compiler source conversion function, is included:
    - (3-1) The left-hand side of the BY phrase in condition 2.
    - (3-2) The right-hand side of the BY phrase in condition 2.

[Y]

1. The pre-compiler source conversion function is used by one of the following methods:
    - (1-1) Compiling with the -CVm option specified for the COBOL command.
    - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
2. A COPY statement with a REPLACING phrase is coded.
3. Pseudo-text is specified on the left-hand side or right-hand side of the BY phrase in condition 2.
4. The pseudo-text in condition 3 contains one or more period separators.
5. The first period separator in condition 4 meets one of the following conditions:
    - (5-1) It is on the left-hand side of the BY phrase and is on a different line than the pseudo-text delimiter on the right side of the pseudo-text.
    - (5-2) It is on the left-hand side of the BY phrase and is on the same line as the pseudo-text delimiter on the right side of the pseudo-text.
    - (5-3) It is on the right-hand side of the BY phrase and is on a different line than the pseudo-text delimiter on the right side of the pseudo-text.

[Impact]

From V13.0.0 or later, for programs that meet condition X, a compilation error might occur in subsequent compilation processing because the pseudo-text is not converted. Alternatively, compilation might complete normally as before, but the text replacement result at compile time might not be as intended.

[Action]

In either case, check the conversion result and the text replacement result at compile time for the locations where the following conversion message is output, and correct the source program appropriately.

- When converting using method (1-1) or (1-2):

```
JMN0501I-W PRCV-m0607W REPLACE statement is described. Convert without performing the REPLACE
statement.
```

- When converting using method (1-3):

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

PH14815

- V/L: V10.1.0 to V12.0.0

Under the following conditions, the COBOL application did not operate correctly during execution. This issue has been fixed.

1. The compiler option INITVALUE is specified, and a value other than 00 is specified as the value of INITVALUE.
2. A data item is defined without an initial value in the working-storage section of the factory data or object data.
3. Reference without setting a value in the data item defined in condition 2.

PH05861

- V/L: V10.0.1 to V11.0.0

Under the following conditions, a COBOL application runtime error occurred. This issue has been fixed.

Symptom: The reference modified range of the sending side shifted one digit to the left.

1. The following MOVE sentences (*) are described.
    - The sending side is a signed zoned decimal item, and
    - The receiving side is a numeric edited data item or a floating point item.
2. The SIGN IS TRAILING SEPARATE CHARACTER clause is specified in the sending data item.
3. There is a reference modification of the sending data item.
4. The length of reference modification in condition 3 is specified with a numeric literal.

*: Includes the implicit MOVE statement.

PH05538

- V/L: V10.1.0 to to V11.0.0

Under the following conditions ([Condition A] to [Condition C]), a COBOL application abnormally terminated (ACCESS VIOLATION) or produced incorrect execution results. This issue has been fixed.

[Condition A]

1. The data item declared to based-storage section is used by either of the following methods.
    - There is a subscript.
      or
    - There is a reference modifier. The high-order-end-character-position of the reference modifier is a variable.
2. The compiler option OPTIMIZE(*) is effective

   *: Default is NOOPTIMIZE.

[Condition B]

1. In the based-storage section, the declared data items are specified in any of the following statements.
    - INSPECT statement
    - STRING statement (Nucleus)

- UNSTRING statement (Nucleus)

[Condition C]

1. In the based-storage section, the data items specified the OCCURS clause are being declared.
2. The data items of 1 are being specified in the INITIALIZE statement.

PH04570

- V/L: V11.0.0

Under the following conditions, either [Symptom A] or [Symptom B] might have occurred during COBOL program compilation. This issue has been fixed.

[Symptom A]

The COBOL compiler terminates normally (*1) without giving an error message.

```
JMN2038I-S Length of the literal in the VALUE clause must not exceed the length of the item. The extra
characters at the right end of the literal are truncated
```

*1: In the created object program, the following literal values are set to the data item of the object program.

- The extra characters at the right end of the literal are truncated to match the item length

[Symptom B]

The COBOL compiler outputs the following error message to a correct syntax.

```
JMN2106I-S The value of the literal following 'THROUGH' in the VALUE clause of the condition-name must
be greater than the value of the literal preceding 'THROUGH'.
```

[Conditions]

1. There is a national item or national edited item that encodes one of the following:
   - UTF32
   - UTF32LE
   - UTF32BE
2. The VALUE phrase is specified as follows.
   - data item in condition 1, or
   - Condition-name of which conditional variable is data item in condition 1
3. In the VALUE phrase in condition 2, the national nonnumeric literal (*2) is specified to meet one of the following.
   - National nonnumeric literal is larger than the sizes specified by the PICTURE phrase
   - The value of the national nonnumeric literal following 'THROUGH' is greater than the value of the national nonnumeric literal preceding 'THROUGH'
4. National nonnumeric literals in condition 3 are from 41 to 80 characters.

*2: The concatenation expression of national nonnumeric literal is contained.

PH02265

- V/L: V10.1.0 to V11.0.0

Under the following conditions, a compiler error (*1) was not output. This issue has been fixed.

*: Compiler messages

```
JMN1775I-S A nonnumeric literal or a national literal must be specified after an AS phrase. The
compiler skips to the next paragraph or division.
JMN1107I-S An invalid character-string is specified in the program-id, or the program-id is missing.
The program-name of the next paragraph or division is used.
```

```
JMN1292I-S An invalid character-string is specified as the program-name, or the program-name is
missing.
JMN5526I-S The method-name specified in the INVOKE statement must be an identifier, nonnumeric
literal, or national character literal. The INVOKE statement is ignored.
JMN5561I-S The method-name specified in an in-line method invocation must be a nonnumeric literal or
a national character literal. The in-line method invocation is ignored.
```

The object program that has been created can be executed correctly.

[Condition A]

1. The concatenation expression is described in the following.
   - AS phrase of program-name
   - AS phrase of class-name
   - AS phrase of method-name
   - AS phrase of property-name
   - AS phrase of program-name-literal
   - Method-name specified for INVOKE statement or in-line method invocation
2. The high order end of concatenation expression in condition 1 is a nonnumeric literal.
3. The concatenation expression in condition 1 contains a hexadecimal nonnumeric literal.
4. The COBOL source program code set and the runtime code set match one of the following combinations:
   a. The COBOL source program code set is SJIS and the runtime code set is SJIS.
   b. The COBOL source program code set is UTF-8 and the runtime code set is Unicode.

[Condition B]

1. The concatenation expression is described in the following.
   - AS phrase of program-name
   - AS phrase of class-name
   - AS phrase of method-name
   - AS phrase of property-name
   - Method-name specified for INVOKE statement or in-line method invocation
2. The high order end of concatenation expression of (1) is a national language nonnumeric literal.
3. The concatenation expression of (1) contains a national hexadecimal nonnumeric literal.
4. The COBOL source program code set and the runtime code set are the following combination:
   a. The COBOL source program code set is SJIS and the runtime code set is SJIS.

Supplementary Explanation:

A literal that specifies in the following must be a nonnumeric literal or a national language nonnumeric literal.

- AS phrase of program-name
- AS phrase of class-name
- AS phrase of method-name
- AS phrase of property-name
- AS phrase of program-name-literal
- Method-name specified for INVOKE statement or in-line method invocation

The above conditions are in violation of the syntax rules.

PG78440

- V/L: V10.1.0

Under the following conditions, the COBOL application produced incorrect execution results. This issue has been fixed.

1. The compiler option BINARY(BYTE) or BINARY(WORD, MLBOFF) is specified.
2. The following intrinsic functions exist:

   [A]

   - FUNCTION MAX
   - FUNCTION MIN
   - FUNCTION MEAN
   - FUNCTION MEDIAN
   - FUNCTION RANGE

   [B]

   - FUNCTION ANNUITY
   - FUNCTION NUMVAL
   - FUNCTION NUMVAL-C
   - FUNCTION RANDOM
3. The arguments of the function in condition 2 above are fixed point of 9 digits or less.
4. If the function in condition 2 is [A], four or more arguments are specified.

PG77383

- V/L: V10.1.0

Under the following conditions, the COBOL application produced incorrect execution results. This issue has been fixed.

1. The compiler option OPTIMIZE is specified (*1).
2. There is a statement that sets packed decimal item to numeric edited data item.
3. The number of digits for integer part of packed decimal item and numeric edited data item are the same and neither both of them doesn't have decimal-part.
4. The numeric edited data item specifies only zero suppression for the edit (Only '9', 'Z', and '*' are used for PICTURE character-string).
5. Before the statement in condition 2 (*2), there is a statement that sets data item (or intermediate result) to packed decimal item.
6. The relation between number of digits for the data item in condition 5 (or intermediate result) and number of digits for the packed decimal part in condition 2 is the following :

```
--------------------------------------------------------
Data item (or intermediate result)    Packed decimal item
---------------------------------------  ---------------
          2                                        3
          4                                        5
          6                                        7
          8                                        9
         10                                       11
         12                                       13
         14                                       15
         16                                       17
------------------------------------------------------
```

7. The data item in condition 5. (or intermediate result) and the packed decimal item in condition 2 don't have the decimal-part.

*1: The default is the compiler option NOOPTIMIZE.

*2: MOVE statement, COMPUTE statement, and so on.

PG76651

- V10.1.0 to V10.4.0

Under the following conditions, the COBOL application produced incorrect results when comparing national items (including national edited items and intrinsic functions) of different lengths during execution. This issue has been fixed.

1. Compiler option RCS (UCS2,LE) or RCS (UTF16,LE) is specified.
2. There is a size comparison of a national item and a national item or national language character constant.
3. At least one side is an item with the ANY LENGTH clause is specified or reference modification.
4. The length of the compared targets differs.

2.2 NetCOBOL Runtime Environment

Information described here is applied to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

2.2.1 Discontinued Visual C++ Redistributable Packages

NetCOBOL V13.0.0 or later does not install the following discontinued Visual C++ Redistributable Packages:

- Microsoft Visual C++ 2005 Redistributable Package
- Microsoft Visual C++ 2008 Redistributable Package
- Microsoft Visual C++ 2010 Redistributable Package
- Microsoft Visual C++ 2013 Redistributable Package

Therefore, executable files created with NetCOBOL V11 or earlier by specifying MSVCRT.LIB at link time may fail to run because the required VC runtime cannot be found. If execution fails, a message similar to the following is output:

For dynamic link structure:

System Error Message:

```
The code execution cannot proceed because MSVCR120.dll was not found. Reinstalling the program may fix
this problem.
```

For dynamic program structure:

NetCOBOL Diagnostic Message:

```
JMW0001I-I
THE ERROR WAS DETECTED IN 'executable file name'.
'the time at which the error occurred'.
'COBOL RUN-TIME MESSAGE'.
'JMP0015I-U[PID:xxxxxxxx TID:xxxxxxxx] CANNOT CALL PROGRAM 'program name'.LOAD=xxxxxxxx CODE=0x7e
PGM=xxxxxxxx LINE=xxxxxxxx'
REPORT FILE:
'file-name'
```

Impact:

If you have executable files created with NetCOBOL V11 or earlier by specifying MSVCRT.LIB at link time, take action.

Verification:

Check whether the executable file's dependent DLLs include the following files:

- MSVCR80.dll
- MSVCR90.dll
- MSVCR100.dll
- MSVCR120.dll

You can check dependent DLLs using the DUMPBIN command (*1) or third-party tools such as Dependencies. If you use tools like Dependencies, download and use them.

*1: Available in the NetCOBOL command prompt of the NetCOBOL Development Package.

```
> dumpbin /DEPENDENTS executable file | findstr /i /c:"MSVCR"
MSVCR120.dll
```

Action:

Relink with NetCOBOL Enterprise Edition Developer (64bit) for Windows V13.

Relink with NetCOBOL Enterprise Edition Developer V13, specifying MSVCRT.LIB during linking. The new MSVCRT.LIB is referenced in NetCOBOL Enterprise Edition Developer V13.

2.2.2 Adding Variable Information for Runtime Messages

Content (JMP0828I-E/U)

Added variable information to runtime messages JMP0828I-E/U.

V12.2 or earlier

```
INVALID VALUE SPECIFIED. $1. LINE=$2. OPD=$3.
```

V13.0 or later

```
INVALID VALUE SPECIFIED. $1. LINE=$2. OPD=$3. VALUE=$4. ADDR=$5. LEN=$6.
```

$4 is set the content of the data stored in the data item.

$5 is set the address of data item.

$6 is set the data item length.

Content (JMP0009I-U,JMP0010I-U)

Added variable information to runtime messages JMP0009I-U and JMP0010I-U.

V12.2 or earlier

```
JMP0009I-U INSUFFICIENT STORAGE AVAILABLE. '$1'
```

```
JMP0010I-U LIBRARY WORK AREA IS BROKEN.
```

V13.0 or later

```
JMP0009I-U INSUFFICIENT STORAGE AVAILABLE. '$1' '$2'
```

$2 is set to the desired library work area length.

```
JMP0010I-U LIBRARY WORK AREA IS BROKEN. BRKADR=0x$1
```

$1 is set to the address of the broken library work area.

2.2.3 Character-Code Conversion Result of Data and Imperfect Character Outside Character-Code Range

Content

The result of the character-code conversion of the data stored in the data item of the DISPLAY statement, the STRING statement (writing 2), the UNSTRING statement (writing 2), and the WRITE statement of the print file changed to prevent an unjustified data error.

- Character-code conversion of data beyond the limits of character-code set conversion origin

    - Prior to V11.0

      It is converted into an alternative character.

    - V11.0 or later

      It becomes a code conversion error when executing it, and the following messages (detailed code 42 (0x2a)) are output.

        - DISPLAY statement: JMP0086I-W
        - STRING statement (writing 2) and the UNSTRING statement (writing 2): An overflow condition has occurred. JMP0260I-U when ON OVERFLOW is not specified
        - WRITE statement of the print file: JMP0310I-I/U and JMP0320I-I/U (It is executed along with the execution result when the input-output error of the file occurs).

- Character-code conversion of imperfect character

  When a surrogate pair of the character-code is a high rank or only one of the subordinate positions is stored or the multi byte character is lacked. The result of converting such an imperfect character is different.

    - V11.0 or earlier

      The character is converted into the replacement character.

    - V11.0 or later

      It becomes a code conversion error during execution, and the following messages (detail code 22 (0x16) or 42 (0x2a)) are output.

        - DISPLAY statement: JMP0086I-W
        - STRING statement (writing 2) and the UNSTRING statement (writing 2): An overflow condition has occurred. JMP0260I-U when ON OVERFLOW is not specified.
        - WRITE statement of the print file: JMP0310I-I/U and JMP0320I-I/U (It is executed along with the execution result when the input-output error of the file occurs).

Action

Correct to store a correct character-code in the data item that refers to a detailed message code when executing it.

Specify the execute environment variable @CBR_CONVERT_CHARACTER=SYSTEM expressly when you want to obtain the result similar to V11.0 or earlier.

However, it is not possible to specify it by the program using the specification V11.0 or later.

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

2.2.4 Replacement Character Used by Code Conversion

Content

The replacement character used has changed when the character-code conversion corresponding to the character-code in the conversion origin does not exist.

- V10.4 or earlier

  Normal-width underscore "_"

- V11.0 or later

  Normal-width question "?"

Action

Specify the execute environment variable information @CBR_CONVERT_CHARACTER=SYSTEM expressly.

However, it is not possible to specify it by the program using the specification V11.0 or later.

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

2.2.5 Change in Severity Code of Message When Application is Executed

Content

The severity code of the message changed when the following applications were executed.

- Prior to V11.0

  JMP0086I-E CHARACTER CODE CONVERSION FAILED. $1 $2

- V11.0 or later

  JMP0086I-W CHARACTER CODE CONVERSION FAILED. $1 $2

Influence

The Return code (PROGRAM-STATUS) of the COBOL program changes with a change in the severity code.

Action

For additional details, refer to the "NetCOBOL Messages."

2.2.6 Interchangeable Information Regarding Bug Fixes

This section describes changes in NetCOBOL runtime environment behavior caused by defect fixes implemented in NetCOBOL V10 or later. V/L indicates the range of versions where the defect existed.

PH22132

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, executing an SQL statement with a data length of 0 for a variable-length string type host variable did not set the host variable data to an empty string. This issue has been fixed.

[Conditions]

1. Use a Windows system.
2. Create the COBOL program as a Unicode application by specifying one of the following compiler options:
   - ENCODE(UTF8,UTF32,LE)
   - ENCODE(UTF8,UTF32,BE)
3. Access the database through the ODBC interface.
4. Set the length field of a national host variable that handles variable-length character string data to 0.
5. Execute an SQL statement with the host variable from condition 4 specified in one of the following locations:
   - An insert value within the insert value list of an INSERT statement.
   - A value specification within the SET clause of an UPDATE statement.
   - A search condition in a WHERE clause.
   - The USING clause of a dynamic SQL statement.

    - The argument list of a CALL statement.

PH15066

- V/L: V12.0.0

Under the following conditions, during the execution of a COBOL application, the result converted to the binary item did not become zero even though the internal floating-point data item of mainframe format was zero. This issue has been fixed.(*1)

*1: Even if it satisfies all of the conditions, this event might not occur according to the state of the memory.

1. The COBOL program is compiled with compiler option FLOAT(M) specified.
2. The internal floating-point data item is converted to the binary item,(*2)
3. The length of internal floating-point data item is 4 bytes or less.
4. The value of internal floating-point data item is zero.

*2: Conversion is done when showing in the following.

- MOVE is done from the internal floating-point data item to the binary data item.
- The receiving data item of an operational result that contains the floating-point data item is the binary data item.

PH15042

- V/L: V12.0.0

Under the following conditions, during the execution of a COBOL application, the sorted records were not correct. This issue has been fixed.

1. PowerBSORT is installed.
2. The COBOL program is compiled with compiler option FLOAT(M) specified.
3. The internal floating-point data item is specified for sort key.
4. The SORT statement is executed with sort key in condition 3.

PH06622

- V/L: V10.1.0 to V11.0.0

Under the following conditions, the record was not printed according to the specified printing attributes or printing position. This issue has been fixed.

For a Unicode application, additionally, the following phenomena occurs. (*1)

- The runtime system message "JMP320I 'CNVER=xx'" is output.
- The printing results of the national item are garbled.

*1: It depends on the data stored in the national item.

1. The print file without FORMAT clause is used.
2. This program is output directly to the printer without using the related product.
3. The record item specified for the WRITE statement or the data item specified for the FROM phrase of WRITE statement is a group item that is subordinate to the data item with the REDEFINES clause.
4. The following either is described in the group item of condition 3 that specifies a REDEFINES clause.
    - There is a group item subordinate.
    - There is a national item subordinate.
    - An effective CHARACTER TYPE clause is specified for the data item subordinate. (*2)
    - An effective PRINTING POSITION clause is specified for the data item subordinate. (*2)

5. After the group item in condition 3, either of the following is described in the same level-number as condition 3.
    - There is an elementary item or a group item that specifies a CHARACTER TYPE clause.
    - There is an elementary item or a group item that specifies a PRINTING POSITION clause.
    - There is a group item that has the data item subordinate with an effective CHARACTER TYPE clause.
    - There is a group item that has the data item subordinate with a PRINTING POSITION clause.

*2: When the REDEFINES clause is specified for either of the following items, during the compilation of a COBOL program, the JMN2224I-W message is outputted. It is a warning that the CHARACTER TYPE clause or the PRINTING POSITION clause specified for the item is invalid.

- The data item with CHARACTER TYPE clause or the PRINTING POSITION clause specified
- The group item that has the subordinate data item with effective CHARACTER TYPE clause or PRINTING POSITION clause

Example program showing what occurs:

```
DATA DIVISION.
WORKING-STORAGE SECTION.
01  DATA1.
  03  DATA2.
    05  DATA31.
      07  DATA31A   PIC X(10).
    05  DATA32  REDEFINES DATA31.
      07  DATA32A.                        *> Condition 4
        09  DATA32A1  PIC X(5).
        09  DATA32A2  PIC X(5).
    05  DATA33.                           *> Condition 5
       07  DATA33A    PIC N(5) MODE-1.


PROCEDURE DIVISION.

    WRITE PRINT-REC FROM DATA1 AFTER PAGE.  *> Condition 3
```

PH03200

- V/L: V11.0.0

Under the following conditions, during the execution of a COBOL application, the record was not written according to the collating sequence of the index key item. This issue has been fixed.

Moreover, the intended record might not be read.

1. When the program is compiled, the compile option ENCODE(SJIS,SJIS) is specified.
2. The item attribute of the prime record key or alternate record key is a national item.
3. The WRITE statement is executed.

PH01026

- V/L: V10.1.0 to V11.0.0

Under the following conditions, during the execution of a COBOL application, the character comparison was not performed correctly. This issue has been fixed.

1. The COBOL program compiled with the compiler option NSPCOMP(ASP) is executed.
2. The data item is encoded in Shift-JIS by specifying any of the following:
    - The compiler option RCS is omitted.
    - The compiler option RCS(SJIS) is specified.
    - The compiler option ENCODE(SJIS,SJIS) is specified.

3. A character comparison is performed under one of the following conditions:
   - A national character comparison with a national data item as an operand
   - A character comparison with a group item as an operand

   However, the following conditions are excluded.
   - Comparison between group items that do not include any national data items
   - Comparison between group items including an item whose attribute does not specify explicit or implicit display
4. The characters being compared in the character comparison described in condition 3 fall within the following character code ranges:
   - X"8181" to X"819F"
   - X"81E0" to X"81FC"
5. One of the operands being compared has a national space (X"8140") immediately following the character position specified in condition 4.
6. The other operand being compared meets one of the following conditions:
   - It has a double-byte ANK space (X"2020") at the same character position as the national space in condition 5.
   - The character position specified in condition 4 is at the end of the data item.

PG97090

- V/L: V10.1.0 to V10.4.0

Under the following conditions, during the execution of a COBOL application, the NATIONAL-OF function did not correctly substitute the alternative character specified in argument -2 when no corresponding national character was fo und for the character converted by the NATIONAL-OF function. This issue has been fixed.

1. The program is compiled with the compiler option RCS(UTF16,BE)..
2. The NATIONAL-OF function is used.
3. Argument-2 is specified for the function in condition 2.
4. Data that is not an alphanumeric character is specified in argument -1 of the function in condition 2, resulting in an internal code conversion error (no corresponding national character is found).

PG79859

- V/L: V10.1.0

Under the following conditions, during the debugging of a COBOL application by the remote debugging facility of NetCOBOL Studio, unexpected messages were output. This issue has been fixed.

```
There is no disk in the drive. Please insert the disk in the drive.
```

1. COBOL application is debugged by remote debugging facility of NetCOBOL Studio.
2. The load module that operates in the process to be diagnosed makes DEBUG information file (*) (PDB file).
3. In the folder where the load module is stored in condition 2, there is no DEBUG information in condition 2.
4. The system that uses NetCOBOL drives the state that cannot be referred by the un-mount of virtual CD drive.
5. The drive character of the drive that makes DEBUG information file in condition 2 and the drive that cannot be referred in condition 4 is corresponding.

* : DEBUG information file (PDB file) is generated when the linkage option "/DEBUG" is specified.

PG79852

- V/L: V10.1.0

Under the following conditions, during the execution of a COBOL application, when an application error or execution time message was generated, an unexpected message was output in the COBOL Error Report before the diagnostic report. This issue has been fixed.

```
There is no disk in the drive. Please insert the disk in the drive.
```

1. Environment variable @CBR_JUSTINTIME_DEBUG=NO is not specified.
2. The load module that operates in the process to be diagnosed makes DEBUG information file (*) (PDB file).
3. In the folder where the load module is stored in condition 2, there is no DEBUG information in condition 2.
4. The system that uses NetCOBOL drives the state that cannot be referred by the un-mount of virtual CD drive.
5. The drive character of the drive that makes DEBUG information file in condition 2 and the drive that cannot be referred in condition 4 is corresponding.

* : DEBUG information file (PDB file) is generated when the linkage option "/DEBUG" is specified.

PG77099

- V/L: V10.1.0

Under the following conditions, the execution environment variable @CBR_SSIN_FILE=THREAD did not take effect, and one input file was shared by file input of the ACCEPT statement within the process. This issue has been fixed. (*)

1. It is an application that operates by multi-thread.
2. The execution environment variable @CBR_SSIN_FILE=THREAD is specified.
3. ACCEPT statement is executed, and data is input from the file.

* : When the execution environment variable @CBR_SSIN_FILE=THREAD is specified, input file of each thread can be opened by file input of ACCEPT statement.

PG76651

- V/L: V10.1.0 to V10.4.0

Under the following conditions, during the execution of a COBOL program, the size comparison of national items varying in length (including national language edited items and those using intrinsic functions) was incorrect. This issue has been fixed.

1. Compiler option RCS(UCS2,LE) or RCS(UTF16,LE) is specified.
2. There is a size comparison of National item and National item or National language character constant.
3. At least one side is an item with the ANY LENGTH clause is specified or reference modification.
4. The length of the compared targets differs.

PG73346

- V/L: V10.1.0

Under the following conditions, during execution, an unnecessary zero string was set in the four lower-order digits of the eight-digit hexadecimal character embedded in the execution time message JMP0320I-I/U. This issue has been fixed.

1. File operation using COBOL file system (*) is done.
2. Either of the following I/O statements is executed.
   - READ statement
   - WRITE statement
   - REWRITE statement
   - DELETE statement
   - START statement
3. The execution of I/O statement in condition 2 failes.

* : This does not apply if high-speed file processing (BSAM) is specified.

2.3 J Adapter Class Generator

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

2.3.1 Code Conversion When Application is Executed

Content

The replacement character used has changed when Shift JIS is specified as code set during execution and data that is out of Shift JIS is included in character strings that are got by using GET-STRING-X method or GET-STRING-N method.

- V10.1 or earlier

  Normal-width underscore "_"

- V11.0 or later

  If the data before conversion is alphanumeric character, normal-width underscore.

  If the data before conversion is national character, double-byte underscore.

Action

Specify the execute environment variable information @CBR_CONVERT_CHARACTER=SYSTEM expressly.

However, it is not possible to specify it by the program using the specification after V11.0.

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

2.3.2 Error Message Output When Code Conversion Error Occurs

Content

In the conversion of data using the java-lang-String class method (such as GET-STRING-X, GET-STRING-N, NEW-STRING-X and NEW-STRING-N), if the data is incorrect, a code conversion error will occur.

Then, in NetCOBOL V11.0 or later, the following error message is to be output.

```
CHARACTER CODE CONVERSION FAILED. ERRNO:DETAIL
```

Action

Modify incorrect data items to store character codes correctly.

2.3.3 Characters that the Results of Code Conversion are Different

Content

In operation of an application with Shift_JIS, some Unicode characters that are passed to Java have changed as follows.

Shift_JIS	Unicode V10.1 or earlier	Unicode V11 or later
8160	U+301C	U+FF5E
8161	U+2016	U+2225
817C	U+2212	U+FF0D
8191	U+00A2	U+FFE0

Shift_JIS	Unicode V10.1 or earlier	Unicode V11 or later
8192	U+00A3	U+FFE1
81CA	U+00AC	U+FFE2

Action

For versions V10.1 or earlier, change the setting by specifying the following environment variable.

```
COBJNI_CONVERT=SJIS
```

2.4 PowerFORM

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

There is no information on interchangeability.

2.5 Fujitsu Mainframe Foating-Point Arithmetic Emulator

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows

There is no information on interchangeability.

2.6 PowerBSORT

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

2.6.1 Interchangeable Information Regarding Bug Fixes

This section describes changes in PowerBSORT behavior caused by defect fixes implemented in PowerBSORT V7 or later.

V/L indicates the range of versions where the defect existed.

PH05442

- V/L: V7.0.0

Under the following conditions, PowerBSORT did not correctly check for the presence of reconstruction or selection fields within records. This issue has been fixed.

As a result of an error in usage or implementation, the output may produce unpredictable or abnormal results.

1. The bsortex command is used.
2. The record format is one of the following.
   - Binary file variable-length record form (-record recform=var), or
   - Text file fixed field specification (-record recform=txtfix)

3. Two output file information options (-output) or more are specified.
4. The following options are specified by two output file information options (-output) or more.
    - Record reconstruction option (-output reconst=...)
    - Record selection option (-output include=.../omit=.../case=...)
5. The field in the record is specified for a reconstruction field for the record reconstruction option (form of "pos.len" or "pos.END").
6. The maximum position of the reconstruction field specified by each output file information option (-output) is different. or,

   The maximum position of the selection field specified by each output file information option (-output) is different.

2.7 Features No Longer Supported

This section describes features and environments that are no longer supported in NetCOBOL V13.0 or later.

2.7.1 Discontinued Features

Describes features no longer provided by each component.

2.7.1.1 NetCOBOL

No features are no longer available.

2.7.1.2 J Adapter Class Generator

No features are no longer available.

2.7.1.3 PowerFORM

No features are no longer available.

2.7.1.4 Fujitsu mainframe format floating-point arithmetic emulator

No features are no longer available.

2.7.1.5 PowerBSORT

Features No Longer Provided in V9.0.0:

- Process definition file option (-P) of the bsort command

  If you want to specify the bsort command options from a file, consider changing to the argument file option (-a option).

2.7.2 Discontinued Components

This section describes the discontinued components.

Components Discontinued in V13.0.0:

- Interstage Studio COBOL plugin

2.7.3 Unsupported Environments

This section describes the unsupported environments.

Environments No Longer Supported in V13.0.0:

- Operating Systems
    - Windows Server 2016

- Windows Server 2012 R2
- Windows Server 2012
- Windows 10
- Windows 8.1

Chapter 3 Program Fix List

This document explains the fix information that was fixed was fixed in this software version and level.

3.1 NetCOBOL Development Environment

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows

V/L indicates the range of versions where the defect existed.

PH16001

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, a COBOL project in NetCOBOL Studio might not be created correctly. As a result, the following symptoms might occur:

1. When "Specify target object" is checked in the context menu of a COBOL source file within the **Source Files** folder of a COBOL project in the **Dependency** view, the target object file is not displayed in the **Target Object Files** folder under the COBOL source file.
2. A build is not executed for a COBOL project exhibiting symptom 1.

[Condition]

One of the following operations is performed:

- Importing a COBOL project by selecting **File** > **Open Projects from File System...** from the menu bar.
- Importing a COBOL project by selecting **File** > Import... from the menu bar.

PH17154

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the following error message is output and conversion fails when converting a COBOL program using the pre-compiler source conversion function:

```
PRCV-ER022U A fatal conversion error has occurred. (module-name=C2GETNAM,detail-code=001)
```

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
2. A COPY statement is coded.
3. The period separator that marks the end of the COPY statement in condition 2 is at the beginning of a line different from the line where the COPY statement is coded.

PH17155

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function incorrectly converts descriptions within the pseudo-text of COPY and REPLACE statements. As a result, one of the following might occur:

[Symptom 1]

An internal contradiction occurs during conversion, resulting in a conversion error or abnormal termination.

[Symptom 2]

An incorrect conversion is performed, and one of the following occurs during subsequent compilation:

- Symptom 2-1: Because text replacement is not performed, compilation completes normally, but incorrect execution results occur when the COBOL program is executed.
- Symptom 2-2: Because text replacement is not performed, a compilation error is output.
- Symptom 2-3: The following compilation errors are output for the COPY or REPLACE statement:

```
JMN1076I-S The COPY statement is invalid, or the separator period is missing.
```

```
JMN1083I-S The REPLACE statement is invalid or a separator period is missing.
```

- Symptom 2-4: Because text replacement is performed with incorrect pseudo-text, a compilation error is output.
- Symptom 2-5: Because text replacement is performed with incorrect pseudo-text, compilation completes normally, but incorrect execution results occur when the COBOL program is executed.

[Symptom 3]

Even if a non-standard conversion is performed, compilation completes normally, and the same execution results as before migration are obtained when the COBOL program is executed.

Note: The text replacement result at compile time can be checked in the source program listing. If a COPY statement is coded, the compiler option COPY must be specified.

[Condition]

Either condition X or condition Y is met.

[X]

1. The pre-compiler source conversion function is used by one of the following methods:
   - (1-1) Compiling with the -CVm option specified for the COBOL command.
   - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - (1-3) Converting using the COBPRECONV command.
2. A REPLACE statement is coded.
3. One of the following, which are targeted for conversion by the pre-compiler source conversion function, is included:
   - (3-1) The left-hand side of the BY phrase in condition 2.
   - (3-2) The right-hand side of the BY phrase in condition 2.

[Y]

1. The pre-compiler source conversion function is used by one of the following methods:
   - (1-1) Compiling with the -CVm option specified for the COBOL command.
   - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
2. A COPY statement with a REPLACING phrase is coded.
3. Pseudo-text is specified on the left-hand side or right-hand side of the BY phrase in condition 2.
4. The pseudo-text in condition 3 contains one or more period separators.
5. The first period separator in condition 4 meets one of the following conditions:
   - (5-1) It is on the left-hand side of the BY phrase and is on a different line than the pseudo-text delimiter on the right side of the pseudo-text.
   - (5-2) It is on the left-hand side of the BY phrase and is on the same line as the pseudo-text delimiter on the right side of the pseudo-text.
   - (5-3) It is on the right-hand side of the BY phrase and is on a different line than the pseudo-text delimiter on the right side of the pseudo-text.

Note: Symptom 1 might occur when condition X is met. If Symptom 1 does not occur, one of the following symptoms will occur for each condition:

- [X]-(3-1): Symptom 2-1, Symptom 2-2, Symptom 2-3 (*), or Symptom 3.
- [X]-(3-2): Symptom 2-3 (*), Symptom 2-4, Symptom 2-5, or Symptom 3.
- [Y]-(5-1): Symptom 2-1 or Symptom 2-2.
- [Y]-(5-2): Symptom 2-3.
- [Y]-(5-3): Symptom 2-4.

*: Symptom 2-3 occurs only when condition [X]-(1-1) or condition [X]- (1-2) is met.

PH17156

- V/L: V12.2.0 to V12.2.0A

When using the pre-compiler source conversion function under the following conditions, one of the following symptoms might occur:

[Symptom 1]

Descriptions targeted for conversion are not converted, resulting in compilation errors in subsequent compilation processing.

[Symptom 2]

Descriptions targeted for conversion are converted incorrectly, resulting in compilation errors in subsequent compilation processing.

[Symptom 3]

Descriptions targeted for conversion are converted incorrectly, resulting in incorrect execution results when the converted COBOL program is executed.

[Symptom 4]

Descriptions not targeted for conversion are converted incorrectly, resulting in compilation errors in subsequent compilation processing.

[Symptom 5]

Descriptions not targeted for conversion are converted incorrectly, resulting in incorrect execution results when the converted COBOL program is executed.

[Symptom 6]

One of the following conversion messages that should be output is not output:

```
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.
```

```
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.
```

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
2. One of the following conditions (a. through d.) is met:
   a. An EVALUATE statement is coded that meets the following conditions:
      - (a1) One of the following selection subjects is specified in a WHEN phrase of the EVALUATE statement:

        (a1-1) Two or more selection subjects are specified using the ALSO phrase.

        (a1-2) A THRU or THROUGH phrase is specified.

- (a2) At least one selection subject in (a1) contains a combination of operands that have the same attributes as those targeted for conversion in a comparison (*).

b. A PERFORM statement is coded that meets the following conditions:

   - (b1) A VARYING phrase is specified.
   - (b2) One or more AFTER phrases are specified after the VARYING phrase.
   - (b3) The conditional expression in the UNTIL phrase of (b1) or (b2) is targeted for conversion in a comparison (*).

c. The following conditions are met:

   - (c1) A PERFORM statement with a VARYING phrase is coded.
   - (c2) One or more AFTER phrases are specified in the PERFORM statement in (c1).
   - (c3) A conditional expression is coded after the PERFORM statement in (c1) in one of the following:

     An IF statement.

     The selection subject of an EVALUATE statement.

     The UNTIL phrase of a PERFORM statement.

     The WHEN phrase of a SEARCH statement.
   - (c4) The conditional expression in (c3) is targeted for conversion in a comparison (*).

d. A PERFORM statement is coded that meets the following conditions:

   - (d1) It is a PERFORM statement with a procedure-name (an out-of-line PERFORM statement).
   - (d2) An UNTIL phrase is specified in the PERFORM statement in (d1).
   - (d3) The conditional expression in the UNTIL phrase in (d2) is targeted for conversion in a comparison (*).
   - (d4) The PERFORM statement in (d1) ends with a period separator.

*: The pre-compiler source conversion function targets conditional expressions containing a combination of operands (including combinations with the left and right operands reversed) that meet one of the following conditions:

- (A) A national item and an alphanumeric item or a zoned decimal item.
- (B) A national item and a nonnumeric literal.
- (C) A national item and the figurative constant ZERO or QUOTE.
- (D) A binary item or a packed decimal item and an alphanumeric item.
- (E) A binary item or a packed decimal item and a nonnumeric literal.

If the operands in (D) or (E) are specified in the UNTIL phrase of a PERFORM statement or the WHEN phrase of a SEARCH statement, a conversion message is output without performing conversion. Therefore, the conversion message that should be output due to this issue might not be output.

Note: One of the following symptoms will occur for each condition:

- Condition a.: Symptom 1, Symptom 2, Symptom 4, or Symptom 5.
- Condition b.: Symptom 1, Symptom 2, or Symptom 6.
- Condition c.: Symptom 1, Symptom 3, or Symptom 6.
- Condition d.: Symptom 1 or Symptom 6.

PH17157

- V/L: V12.2.0 to V12.2.0A

When using the pre-compiler source conversion function under the following conditions, one of the following errors might occur:

```
PRCV-ER116U There is insufficient space to write the work file "$1". Conversion terminated.
```

```
PRCV-ER128S The number of COBOL words in a conditional expression has exceeded the maximum allowed.
Conversion terminated.
```

If error PRCV-ER116U is output, a huge work file is generated until the disk space is exhausted, so the process might appear to hang. If error PRCV-ER116U is output, the work file is deleted automatically. However, manually delete the file if necessary, for example, if the command is forcibly terminated before this error is output.

The work file is output to the following location with the name "XXXXXXXX.WKn" (XXXXXXXX: 8-digit number, n: 1, 2, or 3):

- If the WORKDIR conversion option is specified: The specified folder.
- If the WORKDIR conversion option is not specified:
  - When using the COBPRECONV command: The output folder specified by the -o option.
  - Otherwise: The .preconv folder in the same folder as the pre-conversion source program.

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
2. One of the following conditions is met (*1):
   - The closing parenthesis corresponding to a left parenthesis in a subscript or reference modification within a conditional expression (*2) or MOVE statement is not coded.
   - An imperative statement or a period does not immediately follow a conditional expression (*2).
   - An operand is missing in a conditional expression (*2).
   - The closing parenthesis corresponding to a left parenthesis in an intrinsic function within a conditional expression (*2) is not coded.
   - A COPY statement is not terminated by a period separator.
   - The required word "TO" is not coded in a MOVE statement.

*1: This includes cases where condition 2 is met in a state where text replacement by the source text manipulation function has not been performed.

*2: The target conditional expressions are those coded in one of the following:

- An IF statement.
- The selection subject of an EVALUATE statement.
- The UNTIL phrase of a PERFORM statement.
- The WHEN phrase of a SEARCH statement.

PH17158

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, when converting a COBOL program using the pre-compiler source conversion function, a data item with the GLOBAL clause is generated in the local-storage section. This can result in the following compilation error in subsequent compilation processing:

```
JMN1304I-S The GLOBAL clause can be specified only for a level 01 item in a FILE SECTION, a WORKING-
STORAGE SECTION, a CONSTANT SECTION, or a BASED-STORAGE SECTION.
```

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. A local-storage section header is coded.
3. One of the following conditions is met (*1):
    - A data-name is specified as a file identifier in an ASSIGN clause, but the data-name is not declared.
    - A conditional expression with a combination of operands (*2) that meet one of the following conditions is coded in the selection subject of an IF or EVALUATE statement:
        - A binary item or a packed decimal item and an alphanumeric item.
        - A binary item or a packed decimal item and a nonnumeric literal.

*1: If such a description exists, the pre-compiler source conversion function performs conversion to generate a data item in the working-storage section.

*2: Includes cases where the left and right operands of the combination are reversed.

PH17159

- V/L: V12.2.0 to V12.2.0A

When using the pre-compiler source conversion function under the following conditions, one of the following might occur:

[Symptom 1]

One of the following conversion errors is output:

```
PRCV-ER108S COBOL library '' could not be found.
```

```
PRCV-ER127S The data item name must not be same as the file-identifier. Conversion terminated.
```

```
PRCV-ER126S The duplicate name cannot be specified for files with different ASSIGN clause. Conversion
terminated.
```

[Symptom 2]

Descriptions targeted for conversion are not converted, resulting in compilation errors in subsequent compilation processing.

[Symptom 3]

Descriptions targeted for conversion are converted incorrectly, resulting in compilation errors in subsequent compilation processing.

[Symptom 4]

Descriptions not targeted for conversion are converted incorrectly, resulting in compilation errors in subsequent compilation processing.

[Symptom 5]

Descriptions not targeted for conversion are converted incorrectly, resulting in incorrect execution results when the converted COBOL program is executed.

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. One of the following two-word sequences (*1) is coded across multiple lines:
    - (2a) "ALL" of an ALL literal and a nonnumeric literal.

- (2b) A level number and a data-name.
- (2c) "COPY" and a text-name or text-name-literal in a COPY statement.
- (2d) "PIC", "PICTURE", or "IS" and the PICTURE character-string.
- (2e) "IDENTIFICATION" and "DIVISION" in the Identification Division header, or "END" and "PROGRAM" in the end program header.
- (2f) "CONFIGURATION" and "SECTION" in the configuration section header, or "INPUT-OUTPUT" and "SECTION" in the input-output section header.
- (2g) "FILE" and "SECTION" in the file section header.
- (2h) "PROCEDURE" and "DIVISION" in the Procedure Division header.
- (2i) The name of a split-key (*2) and "=" in an indexed file.

3. The character immediately following the first word in condition 2 is one of the following separators:
    - A space.
    - A comma.
    - A semicolon.

*1: The following example shows a two-word sequence coded across multiple lines:

```
000010 IDENDIFICATION
000020 DIVISION.
000030 PROGRAM-ID.  A.
```

*2: In third-party COBOL, a name called a split key can be declared for the sequence of key items in the RECORD KEY or ALTERNATE RECORD KEY clause of an indexed file. The pre-compiler source conversion function converts descriptions of these split-keys.

Note: The symptom that occurs depends on condition 2, as follows:

- (2a): Symptom 2 (This symptom occurs only when a comparison between an ALL literal that meets the condition and a national item or a national edited item is coded in one of the following):
    - An IF statement.
    - The selection subject of an EVALUATE statement.
    - The UNTIL phrase of a PERFORM statement.
    - The WHEN phrase of a SEARCH statement.
- (2b): The following symptoms occur depending on whether the data item that meets the condition meets the following conditions:
    - If it is a data item targeted for conversion by the pre-compiler source conversion function: Symptom 2.
    - If it is a data item specified in the DYNAMIC phrase of an ASSIGN clause: Symptom 4.
    - If it is a data item with the same name as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause: Symptom 5.
- (2c): Symptom 1.
- (2d): Symptom 2 or Symptom 3 (This symptom occurs only when the data item with the PICTURE clause that meets the condition is targeted for conversion by the pre-compiler source conversion function).
- (2e): The following symptoms occur depending on whether the description that meets the condition meets the following conditions:
    - If it is the Identification Division header for the outermost program: Symptom 3.
    - If multiple programs are declared within the same compilation unit and one of the following conditions is met:
        - If a data item with the same name as a data-name specified in the DYNAMIC phrase of an ASSIGN clause is declared in another program: Symptom 2.
        - If a data item with the same name as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause is declared in another program: Symptom 1.

        - If duplicate names are specified for files in different ASSIGN clauses in different programs: Symptom 1.
        - If multiple data items with the same name are declared within the same compilation unit: Symptom 5.
- (2f): Symptom 2.
- (2g): Symptom 4.
- (2h): Symptom 5 (Although the conversion is designed to be aborted when a data item with the same name as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause is declared, the conversion continues and this symptom occurs when the condition is met).
- (2i): Symptom 2.

PH17160

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the pre-compiler source conversion function performs an incorrect conversion, resulting in one of the following compilation errors in subsequent compilation processing:

```
JMN3242I-S National literal and FUNCTION CAST-ALPHANUMERIC (alphanumeric) cannot be compared.
```

```
JMN2503I-S User word '@1@' is undefined.
```

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. A national nonnumeric literal or a hexadecimal nonnumeric literal is coded.
3. The character immediately preceding the literal in condition 2 is a left parenthesis.
4. There is no space separator between the item in condition 2 and the left parenthesis in condition 3 (*1).
5. The literal in condition 2 is coded in one of the following locations:
    - (5-1) The left-hand side of a conditional expression (*2) coded in one of the following, where the other operand is a national item or a national edited item:
        - An IF statement
        - The selection subject of an EVALUATE statement
        - The UNTIL phrase of a PERFORM statement
        - The WHEN phrase of a SEARCH statement
    - (5-2) On the same line as an item targeted for conversion by the pre-compiler source conversion function (*3)

*1: A single left parenthesis "(" can be used as a separator without spaces around it. This issue occurs when a single left parenthesis is coded immediately followed by a national nonnumeric literal or a hexadecimal nonnumeric literal, without an intervening space.

```
000010       IF (N"あ" = N01) *> Meets the condition
000020       OR ( N"い" = N01) *> Does not meet the condition
```

*2: If the literal in condition 2 is a national nonnumeric literal, an incorrect conversion is performed, resulting in the JMN3242I-S compilation error. If the literal is a hexadecimal nonnumeric literal, the conversion result is correct, and this is not a defect.

*3: The pre-compiler source conversion function inserts line breaks as needed to keep lines within 72 bytes. This issue occurs only when the conversion causes the literal in condition 2 to exceed the 72-byte limit. In the following example, "BEGINNING" is converted to "BEGINNING-RV" because it is a reserved word. However, the inserted line break is in the wrong place. This results in the JMN2503I-S compilation error (User word 'X' is not defined).

(Before conversion)

```
000010*  1---------2---------3---------4---------5---------6---------7--
000020      COMPUTE BEGINNING = 100000000000 + FUNCTION NUMVAL(X"3130").
```

(Correct conversion result)

```
000010*  1---------2---------3---------4---------5---------6---------7--
000020      COMPUTE BEGINNING-RV = 100000000000 + FUNCTION NUMVAL(
000020                                                                  X"3130").
```

(Incorrect conversion result)

```
000010*  1---------2---------3---------4---------5---------6---------7--
000020      COMPUTE BEGINNING-RV = 100000000000 + FUNCTION NUMVAL(X
000020                                                                   "3130").
```

PH17188

- V/L: V12.2.0 to V12.2.0A

When using the pre-compiler source conversion function under the following conditions, one of the following might occur:

[Symptom 1]

One of the following conversion errors is output:

```
PRCV-ER126S The duplicate name cannot be specified for files with different ASSIGN clause. Conversion
terminated.
```

```
PRCV-ER127S The data item name must not be same as the file-identifier. Conversion terminated.
```

[Symptom 2]

Descriptions targeted for conversion are not converted, resulting in compilation errors in subsequent compilation processing.

[Symptom 3]

Descriptions not targeted for conversion are converted, resulting in compilation errors in subsequent compilation processing.

[Symptom 4]

Descriptions not targeted for conversion are converted, resulting in incorrect execution results when the converted COBOL program is executed.

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. Either condition [X] or condition [Y] is met.

    [X]

    - (X1) A program containing an internal program exists.
    - (X2) A file description entry with the GLOBAL clause is declared in the program in (X1).
    - (X3) The level-01 data description entry following the file description entry in (X2) does not have the GLOBAL clause.
    - (X4) A program directly or indirectly contained in the program in (X1) contains one of the following descriptions:
        - (X4-a) The same name as the data item in (X3), or one of its subordinate data items, is specified as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause.

- (X4-b) The data item in (X3), or one of its subordinate data items, is specified in one of the following:

  (X4-b-1) A data-name in the DYNAMIC phrase of an ASSIGN clause.

  (X4-b-2) An operand of a comparison or a move operation targeted by the pre-compiler source conversion function (*1).

[Y]

- (Y1) A program [A] containing two or more internal programs exists.
- (Y2) A data item with the GLOBAL clause exists in program [B], which is directly or indirectly contained in program [A].
- (Y3) A program [C] exists after program [B] and is indirectly contained in program [A].
- (Y4) Program [C] is not contained in program [B].
- (Y5) The hierarchy level (*2) of program [C] is greater than that of program [B].
- (Y6) One of the following conditions is met:
  - (Y6-a) In program [C], the same name as the data item in (Y2), or one of its subordinate data items, is specified as a file identifier specified in the EXTERNAL phrase of an ASSIGN clause.
  - (Y6-b) A data item with the same name as the data item in (Y2), or one of its subordinate data items, exists and meets the following conditions:

    (Y6-b-1) It is declared with the GLOBAL clause in program [A], or a program directly or indirectly containing program [A].

    (Y6-b-2) In program [C], it is specified in one of the following locations:

    (Y6-b-2-1) A data-name in the DYNAMIC phrase of an ASSIGN clause.

    (Y6-b-2-2) An operand of a comparison or a move operation targeted by the pre-compiler source conversion function (*1).

*1: The pre-compiler source conversion function targets comparisons and move operations specified in the following locations:

- A conditional expression in one of the following:
  - An IF statement.
  - The selection subject of an EVALUATE statement.
  - The UNTIL phrase of a PERFORM statement.
  - The WHEN phrase of a SEARCH statement.
- A MOVE statement.

*2: The hierarchy level of a program is a number counted as follows: the outermost program is 1; a program directly contained in the outermost program is 2; a program directly contained in a level-2 program is 3; and so on.

```
Hierarchy level: 1
+--------------------------------------
| Program [A]
| 01 DATA01 PIC X IS GLOBAL.
|Hierarchy level: 2
| +------------------------------------
| | Program [B]
| |  01 DATA01 PIC 9 COMP-5 IS GLOBAL.
| +------------------------------------
|Hierarchy level: 2
| +------------------------------------
| |Hierarchy level: 3
| | +----------------------------------
| | | Program [C]
| | |
| | |  IF DATA01 = "A"
| | |  THEN
| | |
```

Note: The following symptoms occur depending on the condition:

- Condition (X4-a): Symptom 3.
- Condition (X4-b-1): Symptom 4.
- Condition (X4-b-2): Symptom 2, Symptom 3, or Symptom 4.
- Condition (Y6-a): Symptom 1.
- Condition (Y6-b-2-1): Symptom 2.
- Condition (Y6-b-2-2): Symptom 3 or Symptom 4.

PH17604

- V/L: V10.2.0 to V12.2.0A

In NetCOBOL Studio, under the following conditions, the pre-compiler is not executed during a project rebuild, and the pre-compiler output source file generated during the previous build is not updated. Therefore, the pre-compiler output source file generated during the previous build is used for the build.

[Condition]

1. A pre-compiler is added to the build tools (*).
2. An extension other than cob is set for the pre-compiler output source.
3. The pre-compiler input source file is added to the Source Files folder in the Dependencies view.
4. The output source file for the pre-compiler input source file exists.
5. The modification date and time of the output source file is later than that of the input source file.
6. One of the following operations is performed:
    - The project is rebuilt using Manual Build in the build function.
    - The project is cleaned using the Clean Projects dialog without checking "Start a build immediately", and then the project is built using Manual Build or Automatic Build in the build function.
    - The project is rebuilt using Automatic Build in the build function when a COBOL project that satisfies conditions 1 through 5 is imported.

*: If "Use precompiler" is checked on the Target Definition screen of the COBOL Project Generation Wizard, a pre-compiler is automatically added to the build tools.

PH17766

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, when converting a COBOL program using the pre-compiler source conversion function, library text included by a COPY statement written in the syntax specific to third-party COBOL compilers is expanded to an incorrect location. As a result, incorrect execution results might occur when the converted COBOL program is executed.

[Condition]

1. The pre-compiler source conversion function is used by one of the following methods:
    - (1-1) Compiling with the -CVm option specified for the COBOL command.
    - (1-2) Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - (1-3) Converting using the COBPRECONV command.
2. Expansion of library text included by COPY statements into the post-conversion source program is enabled by one of the following:
    - Converting using method (1-1) or (1-2).
    - Converting using method (1-3) with the EXPAND-COPY conversion option set to YES.
3. The pre-conversion source program or library file contains the following description (*):
    - (3-1) A COPY statement with a REPLACING phrase is coded.

- (3-2) A COPY statement is coded within the library text included by the COPY statement in (3-1).

*: The description that satisfies condition 3 is specific to third-party COBOL compilers. When compiling with NetCOBOL without using the pre-compiler source conversion function, the following compiler message is output:

```
JMN1074I-S The COPY statement cannot be specified in library text referenced by a COPY statement with
a REPLACING or JOINING phrase.
```

(Program Example)

Before Conversion:

Pre-conversion source program: A.cob

```
001001       DISPLAY "A START".
001002       COPY B REPLACING DATA01 BY DATA02.
001003       DISPLAY "A END".
```

Library file included by the COPY statement in (3-1): B.cbl

```
002001       DISPLAY "B START".
002002       DISPLAY DATA01.
002003       COPY C.
002004       DISPLAY "B END".
```

Library file included by the COPY statement in (3-2): C.cbl

```
003001       DISPLAY "C START".
003002       DISPLAY DATA01.
003003       DISPLAY "C END".
```

Incorrect Conversion Result:

Post-conversion source program: A.cob

```
001001       DISPLAY "A START".
001002       COPY B REPLACING DATA01 BY DATA02.
003001       DISPLAY "C START".
003002       DISPLAY DATA01.
003003       DISPLAY "C END".
001003       DISPLAY "A END".
```

Post -conversion library file: B.cbl

```
002001       DISPLAY "B START".
002002       DISPLAY DATA01.
002003*      COPY C.
002004       DISPLAY "B END".
```

Program Execution Results:

Before Migration	After Migration
A START	A START
B START	B START
(Value of DATA02)	(Value of DATA02)
C START	B END
(Value of DATA02)	C START
C END	(Value of DATA01)
B END	C END
A END	A END

PH18543

- V/L: V12.0.0 to V12.2.0A

Under the following conditions, imported projects may not appear in the **Dependency** view or **Structure** view:

[Condition]

When importing any of the following projects using the [File] > [Open Projects from File System] menu:

COBOL project

COBOL resource Project

COBOL solution Project

CORBA server Project

*: This issue does not occur when importing projects using [File] > [Import] > [General] > [Existing Projects into Workspace].

PH18625

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, dependency files located in folders specified in the compiler options below are not added when analyzing dependencies after importing a COBOL project in NetCOBOL Studio:

- LIB
- FORMLIB
- REP
- REPIN

[Condition]

1. A COBOL project is imported using either of the following methods:
    - [File] > [Open Projects from File System] from the menu bar.
    - [File] > [Import] from the menu bar.
2. Any of the following compiler options are specified in the project imported in condition 1.:
    - LIB
    - FORMLIB
    - REP
    - REPIN

PH19485

- V/L: V12.0.0 to V12.2.0A

Under the following conditions, the COBOL editor may take a long time to open in NetCOBOL Studio:

[Condition]

1. The reference format for the COBOL editor is set to fixed or variable format.
2. Either of the following is true:
    - The COBOL editor is opened for the first time after starting NetCOBOL Studio.
    - A COBOL source file different from the one opened in the previously active COBOL editor is opened.
3. The COBOL source file opened in condition 2 has a large number of lines (*).

* As a guideline, 10,000 lines or more.

Note: The threshold number of lines might vary depending on hardware specifications and OS conditions.

PH19486

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, opening the Outline view in NetCOBOL Studio might take a long time.

[Condition]

1. Either of the following is true:
    - The Outline view is opened after a COBOL source file is opened in the COBOL editor when the Outline view is closed.
    - A COBOL source file is opened in the COBOL editor when the Outline view is open.
2. The COBOL source file opened in condition 1 has a large number of sections and paragraphs (*).

* As a guideline, 800 or more.

Note: The threshold number of sections and paragraphs might vary depending on hardware specifications and OS conditions.

PH19737

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, the tree structure of the Outline view in NetCOBOL Studio will no longer match the structure of the source code. Depending on the state of the Outline view tree structure, subsequent editing operations might cause the following error:

Message box title: Problem Occurred

Message: Unhandled event loop exception

java.lang.StackOverflowError

[Condition]

1. A source code file is open in the COBOL editor.
2. The Outline view is displayed.
3. The COBOL editor is displayed as a clone, horizontally split, or vertically split.
4. Editing operations are performed in the COBOL editor.

PH19738

- V/L: V10.2.0 to V12.2.0A

Under either of the following conditions, in the NetCOBOL Studio COBOL editor, the correspondence between sequence number lines and source code lines might be misaligned, resulting in a mismatch in the number of lines. When a file is saved with fewer sequence number lines than source code lines, the following error occurs:

- Message box title: Problem Occurred
- Message: Save failed. String index out of range: (number)

[Condition A]

1. The reference format is set to fixed or variable format.
2. A COBOL source file is opened in the COBOL editor.
3. The sequence number area contains non-numeric characters.
4. The following editing operation is performed on the COBOL source file:
    - Lines are moved using drag and drop.
    - An editing operation is undone from the context menu of Quick Diff in the vertical ruler.
5. The editing operation in condition 4 is undone using the undo function of the COBOL editor.
6. A line break is inserted in a line with a non-numeric sequence number described in condition 3.

[Condition B]

1. The reference format is set to fixed or variable format.
2. A COBOL source file is opened in the COBOL editor.
3. The following editing operation is performed on the COBOL source file:
    - Lines are moved using Alt+Up Arrow or Alt+Down Arrow.
    - Multiple lines of code are inserted using content assist.
4. The editing operation in condition 3 is undone using the undo function of the COBOL editor.

PH20814

- V/L: V11.0.1 to V12.2.0A

Under either of the following conditions, the compilation command might not complete.

[Condition A]

Compilation is performed with NetCOBOL V11.0.1 or later and either [X] or [Y] is met.

[X]

1. One of the following national items is coded:
    - The compiler option ENCODE(UTF8,UTF32) is specified and no ENCODING clause is specified.
    - The ENCODING clause specifies an alphabet-name associated with UTF32, UTF32BE, or UTF32LE in the ALPHABET clause.
2. The national item in condition X-1 and a 63-character national nonnumeric literal are coded in one of the following:
    - The receiving or sending item of a MOVE, INITIALIZE, or implied MOVE statement.
    - The conditional variable of a SET statement and the VALUE clause of the condition-name.
    - An operand of a comparison condition in an IF, EVALUATE, PERFORM, or SEARCH statement.
    - An INSPECT, STRING, or UNSTRING statement.

[Y]

1. The compiler option ENCODE(UTF8,UTF32) is specified.
2. A 63-character national nonnumeric literal is coded in one of the following:
    - The USING BY CONTENT phrase of a CALL statement.
    - An argument of the DISPLAY-OF function.

[Condition B]

1. Compilation is performed with NetCOBOL V11.0.1 or later.
2. One of the following items is defined:
    - An alphanumeric edited data item.
    - A national edited data item.
3. A 49- or 50-character string is coded in the PICTURE clause of the item in condition B-2.
4. The item in condition B-2 is specified as the receiving item of one of the following:
    - A MOVE statement.
    - A statement that executes an implied MOVE.

[Program Example for Condition B-2]

```
IDENTIFICATION DIVISION.
PROGRAM-ID. SAMPLE.
DATA DIVISION.
```

```
WORKING-STORAGE SECTION.
01  ED01 PIC /X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/X/.
01  A01  PIC X(24).
PROCEDURE DIVISION.
    MOVE A01 TO ED01.
```

PH21730

- V/L: V10.2.0 to V12.2.0A

When debugging a NetCOBOL application on Linux 64bit using the remote debug function of NetCOBOL Studio, performing a debug operation might result in the error message "This debugging function is not available at present." and the debug operation might fail.

[Environment]

The NetCOBOL for Linux(64) is used for remote debugging.

[Condition]

1. During application debugging, execution is suspended with the message "Abort code: n" (*) displayed in the Console view.
2. Application execution is resumed, and the message "Abort code: 0" is displayed in the Console view.
3. A debug operation, such as resuming execution, is performed.

* When application execution is suspended due to a signal received from the OS, the message "Abort code: n" (n is the signal number) is displayed in the Console view.

PH21990

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, one of the following occurs during COBOL program compilation:

- If condition 3-1 is met, the following message for the COPY statement is incorrectly output, resulting in a compilation error:

  ```
  JMN1075I-S Word COPY must be preceded by a space and followed by a space.
  ```

  The line number embedded in the message indicates the program location where the REPLACE statement in condition 1 is coded.

- If condition 3-2 is met, compilation continues (*) without outputting the message JMN1075I-S for the incorrect COPY statement.

  * If no compilation error occurs, the compilation result of the COPY statement is the same as when one or more spaces are added before the word COPY. However, if the intention is to replace the character string immediately before the character in condition 3-2-2 with the REPLACE statement, the intended result is not achieved because there is no space immediately after the character in condition 3-2-2.

[Example 1]

If one or more spaces exist before the word COPY, "ABC" is correctly replaced with "DEF".

```
000010 REPLACE ==ABC== BY ==DEF==.
000020      DISPLAY ABC, COPY X.
000030 REPLACE OFF.
```

[Example 2]

If no space exists before the word COPY, due to this problem, no compilation error is output for the COPY statement. Because there is no space immediately after the comma, "ABC" in "ABC," is not replaced with "DEF", and compilation continue.

```
000010 REPLACE ==ABC== BY ==DEF==.
000020      DISPLAY ABC,COPY X.
000030 REPLACE OFF.
```

[Condition]

1. A REPLACE statement is coded.
2. A COPY statement is coded within the scope (*1) to be replaced by the REPLACE statement in condition 1.

3. One of the following conditions is met:

   - (3-1) The character position of the REPLACE statement in condition 1 is not at the beginning of a line (*2), and the character position of the COPY statement in condition 2 is at the beginning of a line (*2).

   - (3-2) The character position of the REPLACE statement in condition 1 is at the beginning of a line (*2), and the character position of the COPY statement in condition 2 is not at the beginning of a line (*2). The character immediately preceding the word COPY in condition 2 is one of the following:

     (3-2-1) A separator other than a space, comma, semicolon, or period.

     (3-2-2) A character that is not a separator: a comma, semicolon, or period.

4. The compiler option NONUMBER (*3) is in effect.

*1: Until the next REPLACE statement appears or the end of the compilation unit.

*2: The beginning of a line refers to the following character positions, depending on the reference format:

- Fixed or variable format: column 8

- Free format: column 1

*3: The default value of the compiler option is NONUMBER.

PH22148

- V/L: V10.1.0 to V12.2.0A

Under the following condition, during COBOL program compilation, content coded after the non-target character position in a source program might be treated as if it were coded on the next line and included in the compilation target. This can result in incorrect compilation errors.

Note: Because the content after the specified byte count on the line is treated as if it were coded on the next line, the type of compilation error cannot be specified.

[Condition]

A line with 256 bytes or more (*) is coded in a library text file.

* This includes cases where the target line is a comment line.

PH22316

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, the compiler might abnormally terminate during COBOL program compilation, outputting the following message:

```
JMN0102I-U The compilation process cannot be continued. If other diagnostic messages have been
generated, correct those errors and try the compilation again. (substep-name=JMN300, module-
name=SC30XCNM, detail-code=3037, line-number=8)
```

[Condition]

1. A comparison condition is coded where the left operand is the figurative constant ZERO.

2. The right operand of the comparison condition in condition 1 is an intrinsic function.

3. The intrinsic function in condition 2 meets one of the following conditions:

   - The argument of the FACTORIAL function is an integer numeric literal from 0 through 19.

   - The argument of the INTEGER function is a fixed-point numeric literal.

   - The argument of the INTEGER-PART function is a fixed-point numeric literal.

   - The argument of the LENG function is a fixed-length data item.

   - The argument of the LENGTH function is a fixed-length data item.

   - The argument of the MAX function is one fixed-point numeric literal.

- The argument of the MEDIAN function is one fixed-point numeric literal.
- The argument of the MIDRANGE function is one or two fixed-point numeric literals.
- The argument of the MIN function is one fixed-point numeric literal.
- The argument of the NUMVAL function is a valid nonnumeric literal.
- The argument of the NUMVAL-C function is a valid nonnumeric literal.
- The arguments of the RANGE function are two fixed-point numeric literals, and the sum of their maximum integer-part digits and maximum decimal-part digits does not exceed 18.
- The argument of the SUM function is one fixed-point numeric literal, or the arguments of the SUM function are two fixed-point numeric literals, and the sum of their maximum integer-part digits and maximum decimal-part digits does not exceed 30.

PH22365

- V/L: V12.2.0 to V12.2.0A

Under the following condition, when converting a COBOL program using the pre-compiler source conversion function, the sequence of key items for the first defined split-key (*) is always output for the split-key specified in a READ or START statement for an indexed file. As a result, the execution results of the READ or START statement for the indexed file in the converted COBOL program might differ from those of the pre-migration program.

*: In third-party COBOL, a name called a split key can be declared for the sequence of key items in the RECORD KEY or ALTERNATE RECORD KEY clause of an indexed file.

The pre-compiler source conversion function converts split-keys specified in READ or START statements into the corresponding sequence of key items.

[Example of a Program in third-party COBOL Compiler]

```
     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. *> Equivalent to coding "K1 K2" in third-party COBOL compilers
             READ IXFILE
             KEY IS RKEY2. *> Equivalent to coding "A1 A2" in third-party COBOL compilers
```

[Correct Conversion Result]

```
    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.
       KEY IS = K1 K2.
       READ IXFILE
*      KEY IS RKEY2.
       KEY IS A1 A2.
```

[Incorrect Conversion Result]

```
     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.
       KEY IS = K1 K2.
       READ IXFILE
*      KEY IS RKEY2.
       KEY IS K1 K2. *> Incorrect code
```

[Condition]

1. The pre-compilation source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
    - Converting using the COBPRECONV command.
2. An indexed file with two or more split keys defined is declared.
3. A READ or START statement is coded for the file in condition 2.
4. A split key defined second or later is specified in the KEY phrase of the statement in condition 3.

PH22368

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the following problems occur with the object comparison tool (COBCMPO command) and the COBOL compiler:

- During COBCMPO command execution:
    - (A-1) The COBCMPO command results in a segmentation fault or an infinite loop.
    - (A-2) The COBCMPO command terminates with the following message:

```
"xxxx: Different Program name." (xxxx is the file name)
```

- During COBOL program compilation:
    - (B-1) The COBOL compiler results in a segmentation fault or an infinite loop.
    - (B-2) The COBOL compiler outputs an object file even if there is no difference compared to the object file in the output destination.

[Condition]

- During COBCMPO command execution:
    1. The object file (*1) specified for the COBCMPO command is compiled with the compiler option SCS(UTF8).
    2. The program-name (or class-name, method-name) of the source program from which the object file in condition 1 is compiled has an AS phrase.
    3. A national nonnumeric literal is coded in the AS phrase in condition 2.
    4. The number of characters in the national nonnumeric literal in condition 3 exceeds 53 (*2).
- During COBOL program compilation:
    1. The compiler option OBJECT(DIFF) and SCS(UTF8) are specified.
    2. The program-name (or class-name, method-name) has an AS phrase.
    3. A national nonnumeric literal is coded in the AS phrase in condition 2.
    4. The number of characters in the national nonnumeric literal in condition 3 exceeds 53 (*2).

*1: If a folder is specified, the object files under the folder are targeted.

*2: Problem A-2 or B-2 occurs.

[Program Example]

```
   PROGRAM-ID. A AS N"１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０
 -                   "１２３４５６７８９０".
```

PH22430

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, one of the following compiler messages is output during COBOL program compilation:

```
JMN2880I-S The number of characters for the host variable defined as a national item cannot exceed
16346.
```

```
JMN2886I-S The number of characters for a national item that is subordinate to a host variable of
variable-length string type cannot exceed 16344 characters.
```

[Condition]

1. The compiler option ENCODE(UTF8,UTF32) is specified.
2. One of the following host variables is coded:
    - A host variable for a national item that exceeds 8173 characters.
    - A host variable of variable-length character string type that has a subordinate national item that exceeds 8172 characters.

Note:

The character count of a host variable for a national item encoded in UTF-32 must be 8173 characters or less for fixed length and 8172 characters or less for variable length.

PH22556

- V/L: V10.1.0 to V12.2.0A

Under the following condition, the following compiler message is incorrectly output during COBOL program compilation, resulting in a compilation error:

```
JMN3366I-S An integer function and a numeric function cannot be specified in the SET statement.
```

[Condition]

1. One of the following SET statements is coded:
    - A format-1 SET statement where an index-name is specified as the receiving operand.
    - A format-2 SET statement.
2. The LENG function is specified as the sending operand in the SET statement in condition 1.

Note:

The LENG function can be coded wherever an integer numeric literal can be coded in the PROCEDURE DIVISION. Compilation should be allowed for the SET statements described in the condition.

PH22571

- V/L: V10.1.0 to V12.2.0A

Under the following condition, the maximum record length value in the following compiler message is incorrect:

```
JMN2247I-S The record length of a sequential file with a variable length format cannot exceed 32752
bytes. A record length of 32752 bytes is assumed.
```

[Condition]

1. One of the following files is described:
   - (A) A sequentially organized file (*1) that does not meet any of conditions (A-1) through (A-3):

     (A-1) The file identifier is in one of the following formats:
     - [DA-][nn-]VS-name (nn is a two-digit number)
     - [comment-]AS-name

     (A-2) A print file with FORMAT clause.

     (A-3) A presentation file (*2).
   - (B) A line sequentially organized file (*1).
   - (C) A sort-merge file (*3).
2. The record format of the file in condition 1 is variable length (*4).
3. The maximum record length (*4) of the file in condition 1 exceeds 32756 bytes.

*1: For file organization, refer to "File Organization" in "NetCOBOL Language Reference."

*2: For presentation files, refer to "Presentation File Module" in "NetCOBOL Language Reference."

*3: For sort-merge files, refer to "Sort-Merge Module" in "NetCOBOL Language Reference."

*4: For record format and maximum record length, refer to "Record Format" in "NetCOBOL Language Reference."

PH22600

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, when the value of a national item encoded in UTF-32 is referenced during debugging using the NetCOBOL Studio debug function, en-size kana characters are displayed as "?".

[Conditions]

1. A COBOL application is being debugged using the NetCOBOL Studio debug function.
2. One of the following national items is coded:
   - The compiler option ENCODE(UTF8,UTF32) is in effect, and no ENCODING clause is specified.
   - The ENCODING clause specifies an alphabet-name associated with UTF32, UTF32BE, or UTF32LE in the ALPHABET clause.
3. En-size kana characters are set as the value of the national item.
4. The value of the national item is displayed using one of the following NetCOBOL Studio functions:
   - Tooltips in the COBOL editor
   - The Watch view

PH22693

- V/L: V10.1.0 to V12.2.0A

Under any of the following conditions, the line number displayed in messages output by the COBOL runtime system might not be the line number of the statement where the error was detected:

[Condition A]

A complex condition using AND or OR is coded.

[Condition B]

One of the following statements is coded:

- An EVALUATE statement with an ALSO phrase.
- A PERFORM statement with a VARYING phrase.
- A SEARCH statement with a VARYING phrase.

[Condition C]

On the same line, one of the following statements is coded, followed by one or more verbs:

- An arithmetic statement with an ON SIZE ERROR or NOT ON SIZE ERROR phrase.
- A CALL statement that calls an internal program.
- An EVALUATE statement.
- An IF statement.
- A PERFORM statement.
- A MERGE statement with an OUTPUT PROCEDURE phrase.
- A SEARCH statement.
- A SORT statement with an INPUT PROCEDURE or OUTPUT PROCEDURE phrase.
- A STRING statement with an ON OVERFLOW or NOT ON OVERFLOW phrase.
- An UNSTRING statement with an ON OVERFLOW or NOT ON OVERFLOW phrase.
- A RAISE statement.
- An INVOKE statement with a RAISING phrase.
- An input-output statement for which an error handling procedure is specified by a USE statement.

[Program Examples]

Example 1:

```
IF I = 1 THEN DISPLAY "A" ELSE DISPLAY "B" END-IF.
```

Example 2:

```
DISPLAY "A". PERFORM P1. DISPLAY
  "B".
```

PH23189

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, the following problems might occur:

- During COBOL program compilation:

  The replacement of object files by the compiler option OBJECT(DIFF) (*) does not work correctly, and an old object file is linked, which might result in incorrect execution results. If the object file is not replaced, the following compiler message is output:

  ```
  JMN6417I-I Object program has the same binary as an existing file in the output destination. Object
  program is not written.
  ```

  * The compiler option OBJECT(DIFF) enables the function to prevent object file replacement if there is no difference between the generated object file and the object file with the same name in the output destination.

- During COBCMPO command execution:

  When comparing object files using the COBCMPO command, the result is "Match" even if the contents of the object files differ.

[Conditions]

- During COBOL program compilation:
    1. The compiler option DATAAREA(STATIC) and OBJECT(DIFF) are specified.
    2. An object file with the same name exists in the output destination.
- During COBCMPO command execution:

  Object files generated with the compiler option DATAAREA(STATIC) are being compared.

PH23738

- V/L: V10.1.0 to V12.2.0A

Under any of the following conditions in the specified environment, one or more of the following problems might occur:

[Symptoms]

- At compile time, incorrect messages (JMN2112I-S, JMN2113I-S, JMN2245I-S, JMN2247I-S, JMN2832I-S, JMN5400I-U) are output and the object program is not generated.
- At compile time, an incorrect message (JMN2343I-W) is output.
- At compile time, a necessary message (JMN2113I-S) is not output and the object program is generated.
- At runtime, a file with an incorrect format is generated.
- At runtime, a file open error (JMP0310I-I/U) occurs.
- At runtime, incorrect results are obtained when reading a file.
- At runtime, an error (JMP0019I-U, JMP0022I-U) occurs for an EXTERNAL file or data, and the program terminates abnormally.

[Condition A]

A record description entry is coded that meets the following conditions:

1. It has a subordinate group item.
2. The subordinate group item has a subordinate item with a TYPE clause.
3. The group item in condition 2 is redefined.

Example:

```
01 A.
   02 A-1.                 *> Corresponds to condition 2.
      03 A-1-1 TYPE T-1.
   02 A-2 REDEFINES A-1. *> Corresponds to condition 3.
      03 A-2-1 TYPE T-2.
      03 A-2-2 PIC X(8).
```

[Condition B]

A file description entry or a sort-merge file description entry is coded that meets the following conditions:

1. The record description entry associated with the file has a subordinate item with a TYPE clause.
2. The record description entry in condition 1. also meets one of the following conditions:
    - It is also has a subordinate item without a TYPE clause.
    - It meets the conditions of [Condition A].

Example:

```
FD B1.
01 B1-1.
   02 B1-1-1 TYPE T-1.
   02 B1-1-2 PIC X(8).
FD B2.
01 B2-2.     > Corresponds to condition 1 and condition 2(Condition A).
```

```
   02 B2-2-1.
      03 B2-2-1-1 TYPE T-1.
   02 B2-2-2 REDEFINES B2-2-1.
      03 B2-2-2-1 TYPE T-2.
      03 B2-2-2-2 TYPE T-3.
```

[Condition C]

A file description entry or a sort-merge file description entry is coded that meets the following conditions:

1. Two or more record description entries are coded for the file.
2. At least one of the record description entries in condition 1. meets both of the following conditions:
    - It does not have any subordinate elementary items without a TYPE clause.
    - It does not meet the conditions of [Condition A].
3. At least one of the record description entries in 1. meets one of the following conditions:
    - It is an elementary item without a TYPE clause.
    - It is a group item that does not have any subordinate elementary items with a TYPE clause.

Example:

```
FD C.
01 C-1.                    *> Corresponds to condition 2.
   02 C-1-1 TYPE T-1.
   02 C-1-2.
      03 C-1-2-1 TYPE T-2.
      03 C-1-2-2 TYPE T-2.
01 C-2.
   02 C-2-1 PIC X(8).
   02 C-2-2 TYPE T-3.
01 C-3 PIC X(8).         *> Corresponds to condition 3.
01 C-4.                    *> Corresponds to condition 3.
   02 C-4-1 PIC X(8).
   02 C-4-2 PIC X(8).
```

PH23739

- V/L: V10.1.0 to V12.2.0A

Under the following condition, the following compiler message might be incorrectly output during program compilation:

```
JMN2128I-S The level 01 item must not exceed 2147483647 bytes in length.
```

[Condition]

A level-01 group item meeting the following conditions is coded:

1. The item length is 2147483647 bytes or less.
2. The group item has a subordinate group item.
3. The group item in condition 2 has a subordinate item with a TYPE clause.
4. The group item in condition 3 is redefined.

Example:

```
01  A.
    02  A-1.             *> Corresponds to condition 3
        03  TYPE T-1.
    02  REDEFINES A-1. *> Corresponds to condition 4
        03  PIC X(16).
    02  PIC X(2147483631).
01  T-1 TYPEDEF PIC X(16).
```

PH23923

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, the error "User ID or password is incorrect." is displayed when using the remote development function of NetCOBOL Studio.

[Conditions]

1. The password for the remote server contains the "%" character.
2. The "Save user name and password" check box is selected in the New(or Modify) Server Information dialog box.

PH23981

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, one of the following problems occurs:

- If condition 4 is [A]:

  During compilation, incorrect compiler messages (JMN5789I-S, JMN5791I-S) are output, and the object program is not generated.

- If condition 4 is [B]:

  During execution, one of the following problems occurs:

  - A line sequential file with mixed encoding form is generated.
  - An error (JMP0310I-I/U ACC-METHOD) is output by the OPEN statement for an existing line sequential file.
  - When reading a line sequential file, invalid NULL characters are set in the record, or a line feed code is read as data without being recognized as a record delimiter.

  A program that contains multiple national encoding forms in the record associated with the file and in the FROM phrase of a WRITE statement does not result in a compilation error, and the above problems occur.

- If condition 4 is [C]:

  An incorrect compiler message (JMN3565I-S) is output during compilation, or one of the following problems occurs during execution:

  - Print results or screen display shows garbled characters.
  - The endian of data input from the screen is reversed.
  - A WRITE statement error (JMP0320I-I/U ERRCD=9022 or 9032) is output during printing.
  - An open error (JMP0310I-I/U UNSUPPORT) is output during screen display.

  A program that contains multiple national encoding forms in the record associated with the file and in the FROM phrase of a WRITE statement does not result in a compilation error, and the above problems occur.

- If condition 4 is [D]:

  An incorrect program does not result in a compilation error, and one of the following problems occurs during execution:

  - A line sequential file with mixed encoding form is generated.
  - Print results or screen display shows garbled characters.
  - A WRITE statement error (JMP0320I-I/U ERRCD=9022 or 9032) is output during printing.

[Conditions]

1. One of the following is in effect for the compiler option ENCODE (*1):

   - ENCODE(UTF8)
   - ENCODE(UTF8,UTF16)
   - ENCODE(UTF8,UTF32)
   - ENCODE(UTF8,UTF16,LE)

    - ENCODE(UTF8,UTF16,BE)
    - ENCODE(UTF8,UTF32,LE)
    - ENCODE(UTF8,UTF32,BE)
2. An alphabet-name corresponding to one of the following is declared in the ALPHABET clause:
    - UTF16
    - UTF16LE
    - UTF16BE
    - UTF32
    - UTF32LE
    - UTF32BE
3. The alphabet-name from condition 2 is specified in the ENCODING clause.
4. One of the following file declarations ([A] to [D]) is made:

[A]

- A-1. A print file without FORMAT clause or a line sequential file is declared.
- A-2 A record description entry associated with the file in A-1 contains a data description entry with an ENCODING clause that satisfies condition 3.
- A-3. The data description entry in A-2 is a national item or contains a national item.
- A-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in A-1.

[B]

- B-1. A line sequential file is declared.
- B-2. The record description entry associated with the file in B-1 contains only one elementary item.
- B-3. The data description entry in B-2 has an ENCODING clause that satisfies condition 3.
- B-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in B-1.

[C]

- C-1. A print file with FORMAT clause, or a presentation file where a display or printer is specified in the destination type, is declared.
- C-2. A record description entry associated with the file in C-1 contains a data description entry with an ENCODING clause that satisfies condition 3 (*2).
- C-3. The data description entry in C-2 is a national item or contains a national item.
- C-4. The encoding form specified in condition 3 does not match the encoding form explicitly or implicitly specified in the file description entry of the file in C-1.

[D]

- D-1. A line sequential file or a presentation file where display or printer is specified in the destination type is declared.
- D-2. A WRITE statement with a FROM phrase is coded for the file in D-1.
- D-3. The data item specified in the FROM phrase of the WRITE statement in D-2 is a national item or contains a national item.
- D-4. An ENCODING clause that satisfies condition 3 is specified in either of the following:
    - (a) The data description entry specified in the FROM phrase of the WRITE statement in D-2 (*2)
    - (b) The file description entry of the file in D-1
- D-5. The encoding form in (a) and (b) are different.

*1: The compiler option ENCODE shown in condition 1 is in effect when the compiler option is specified, or when the compiler option RCS(UTF16) or RCS(UCS2) is specified.

*2: Includes cases where the ENCODING clause is specified in a COPY statement with IN/OF XMDLIB.

PH24048

- V/L: V10.1.0 to V12.2.0A

Under the following condition, compilation of a COBOL program might terminate with one of the following messages:

```
JMN0100I-U A fatal compiler error has occurred. (substep-name=(*1), module-name=(*1), detail-
code=5480, line-number=nnnn)
```

or

```
JMN0102I-U The compilation process cannot be continued. If other diagnostic messages have been
generated, correct those errors and try the compilation again. (substep-name=(*1), module-name=(*1),
detail-code=5480, line-number=nnnn)
```

*1: The substep-name and module-name vary depending on the system.

[Condition]

The compiler option OPTIMIZE is specified.

PH24197

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, when converting a COBOL program using the pre-compiler source conversion function, one of the following might occur:

- If condition 2a-1 is met:

  Descriptions targeted for conversion are not converted, resulting in compilation errors in subsequent compilation processing.

- If condition 2a-2 is met:

  An incorrect conversion is performed, resulting in compilation errors in subsequent compilation processing.

- If condition 2a-3 is met:

  An incorrect conversion is performed, or descriptions targeted for conversion are not converted, resulting in compilation errors in subsequent compilation processing.

- If condition 2b is met:

  The command will terminate abnormally.

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:

   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.

2. One of the following conditions is met:

   - (2a) A user-defined word exceeding 30 bytes is coded in one of the following:

     (2a-1) An operand of a comparison or a move operation targeted by the pre-compiler source conversion function (*1).

     (2a-2) A data-name coded in the DYNAMIC phrase of an ASSIGN clause (*2).

     (2a-3) One of the following user-defined words in an indexed file with a split-key (*3) specified:

     - (2a-3-1) file name
     - (2a-3-2) split key name

        - (2a-3-3) split key element name

    - (2b) A user-defined word meeting the following conditions is coded:

      (2b-1) It does not contain em-size characters.

      (2b-2) It contains alphabetic characters or en-size kana characters.

      (2b-3) The character at the 33rd byte position is "1".

*1: The pre-compiler source conversion function targets comparisons and move operations specified in the following locations:

- A conditional expression coded in one of the following:
    - An IF statement
    - The selection subject of an EVALUATE statement
    - The UNTIL phrase of a PERFORM statement
    - The WHEN phrase of a SEARCH statement
- A MOVE statement

*2: In third-party COBOL compilers, the following words can be coded after the ASSIGN clause to specify how a physical file name is assigned:

- DYNAMIC: Indicates that the following name is a data-name.
- EXTERNAL: Indicates that the following name is an environment variable name.

If these words are omitted, DYNAMIC is assumed.

*3: In third-party COBOL, a name called a split key can be declared for the sequence of key items in the RECORD KEY or ALTERNATE RECORD KEY clause of an indexed file. The pre-compiler source conversion function converts descriptions of these split keys. A split key element name refers to the data-name of the key item specified in the split key.

PH24198

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, when compiling a COBOL program using the pre-compiler source conversion function, one of the following compiler messages might be output:

```
JMN2500I-S The compiler has encountered a word '@1@' where a statement is expected.
```

```
JMN2672I-S The @1@ phrase contains no valid operand.
```

```
JMN3482I-S @1@ defined in the LINKAGE SECTION must be specified in a USING phrase or a RETURNING
phrase of the PROCEDURE DIVISION, or in a USING phrase of an ENTRY statement.
```

[Conditions]

1. The pre-compiler source conversion function is used by one of the following methods:
    - Compiling with the -CVm option specified for the COBOL command.
    - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
2. A COPY statement is coded in the PROCEDURE DIVISION header.

PH24202

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, when converting a COBOL program using the pre-compiler source conversion function, the following conversion error might be output, or descriptions that should be converted might not be converted, resulting in compilation errors in subsequent compilation processing:

```
PRCV-ER108S COBOL library 'XXXX' could not be found.
```

[Conditions]

1. The system locale of the Windows system is not Japanese.
2. The pre-compiler source conversion function is used by one of the following methods:
   - Compiling with the -CVm option specified for the COBOL command.
   - Compiling with the compiler option PRECONV(MF) specified in NetCOBOL Studio.
   - Converting using the COBPRECONV command.
3. The following conditions are met:
   - (3-1) The compiler option SCS(ACP) is in effect (*).
   - (3-2) A character in the code ranges 0x81 - 0x9F or 0xE0 - 0xFC is coded at the end of a nonnumeric literal.

* The compiler option SCS(ACP) is enabled by default.

PH24251

- V/L: V10.1.0 to V12.2.0A

When performing remote debugging using NetCOBOL Studio or the interactive debugger, the memory usage of the remote debugger connector (*) increases.

* There are remote debugger connectors that run on the client and on the server.

The client-side remote debugger connector command is cobrdc32.

The server-side remote debugger connector commands are as follows:

- NetCOBOL for Windows(32): cobrds32
- NetCOBOL for Windows(64): cobrds64
- NetCOBOL for Solaris or NetCOBOL for Linux(64): svdrds

[Conditions]

1. The remote debugger connector is started.
2. Remote debugging is started, and a connection to the remote debugger connector is established.

PH24348

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, the INSDBINF command might abnormally terminate during execution:

[Conditions]

1. The INSDBINF command is used.
2. The original source program contains one of the following lines:
   - A line whose byte count, excluding the line feed character, is 530 bytes or more, and whose line feed character is CRLF.
   - A line whose byte count, excluding the line feed character, is 531 bytes or more, and whose line feed character is LF.

Note:

- The original source program is the input file for the INSDBINF command and refers to the source program containing embedded SQL statements.
- The INSDBINF command does not support original source programs with lines exceeding 530 bytes, excluding the line feed character. If the command does not abnormally terminate, it outputs the following message (xxxx.pco: original source program name, nn: line number) and terminates with an error:

```
xxxx.pco nn: The file record length is too long.
```

PH24568

- V/L: V10.4.0 to V12.2.0A

Under the following conditions, the template added or updated by the template function of NetCOBOL Studio is not reflected in the content assist list of the COBOL editor.

[Conditions]

1. A new template is added or an existing template is updated using the template function of NetCOBOL Studio.
2. The content assist list is displayed in the COBOL editor.

PH24569

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, one of the following symptoms might occur in NetCOBOL Studio:

Symptom 1: Text annotations, such as errors or search results, might be displayed in shifted positions.

Symptom 2 :Unexpected error messages (*) might be displayed.

Symptom 3: When printing source code, an area different from the specified range might be output.

* Error message details:

Message box title: Problem Occurred

Message: 'File Search' has encountered a problem.

Clicking [Details] displays "java.lang.NullPointerException".

[Conditions for Symptom 1]

When any of the following conditions are met:

[1-A]

1. The reference format is set to fixed or variable format.
2. A file containing error, warning, or search result annotations is opened in the COBOL editor.
3. A line with a length less than 6 columns exists.
4. Error, warning, or search result annotations exist after the line in condition 3.

[1-B]

1. The reference format is set to fixed or variable format.
2. A COBOL program is built remotely.
3. A COBOL search is executed.
4. A file containing search result annotations is opened in the COBOL editor.
5. Text preceding the search result annotations is edited.
6. The source file is saved and closed.
7. The source file is reopened.

[1-C]

1. Annotations such as errors, warnings, search results, or bookmarks exist in the COBOL editor.
2. The reference format setting is changed.

[Conditions for Symptom 2]

When any of the following conditions are met:

[2-A]

1. The reference format is set to free format.

2. A file is opened and edited in the COBOL editor.
3. A COBOL search is executed.

[2-B]

1. A file is opened in the COBOL editor.
2. A COBOL search is executed using a character string that exists in the file opened in condition 1.
3. The reference format setting is changed.
4. A COBOL search is executed.

[Conditions for Symptom 3]

1. The reference format is set to fixed or variable format.
2. A file is opened in the COBOL editor.
3. Multiple lines of source code are selected using the mouse or keyboard.
4. Print is executed from the File menu.
5. In the displayed Print dialog box, Selection is selected in Page Range, and printing is performed.

PH24599

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, changing the compilation order in the NetCOBOL Studio Dependency view may cause a delay before the workbench becomes operable.

[Condition]

1. A large number of object-oriented COBOL source files are added to the Source Files folder in the NetCOBOL Studio Dependency view (*).
2. The Repository paragraph of source files describes dependent classes.
3. Repository files (.rep files) are added to the Dependent Files folder as a result of dependency analysis in the [Dependency] view.
4. The compilation order of the source files is changed using the change compilation order function in the Dependency view.

*: It depends on machine specifications and the number of repository files added to the Dependent Files folder. If the number of source files exceeds 200, it may take more than ten seconds for the workbench to become operable after changing the compilation order.

3.2 NetCOBOL Runtime Environment

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

V/L indicates the range of versions where the defect existed.

PH20024

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, either Symptom 1 or Symptom 2 occurs:

[Symptom 1]

The print position (X-coordinate) of data items for which the CHARACTER TYPE clause or PRINTING POSITION clause is effective is output further towards the end of the line than intended. Furthermore, data items may overlap.

[Symptom 2]

The print position (X-coordinate) of data items is output further towards the end of the line than intended.

[Condition]

1. Use a print file without FORMAT clause.
2. Specify a printer as the output destination.
3. Specify LP as the print format in the I control record or the print information file.
4. Output a line record using a WRITE statement that contains a mixture of data items for which the CHARACTER TYPE clause or PRINTING POSITION clause is effective and data items for which it is not effective.
5. Output a data item for which the CHARACTER TYPE clause or PRINTING POSITION clause is effective immediately after a data item for which neither clause is effective.

The difference between Symptom 1 and Symptom 2 depends on the value specified for the environment variable information @CBR_PrintTextPosition (Specify method of calculating character arrangement coordinates).

a. Specify TYPE2 (with correction processing) or do not specify the environment variable information @CBR_PrintTextPosition. (Note) This results in Symptom 1.
b. Specify TYPE1 (without correction processing). This results in Symptom 2.

Note: If the environment variable information @CBR_PrintTextPosition is omitted, TYPE2 (with correction processing) is the default value.

PH21095

- V/L: V12.0.0 to V12.2.0A

Under the following environment and conditions, either Symptom 1 or Symptom 2 occurs:

[Symptom 1]

Under Condition A, the execution of a SORT statement or MERGE statement results in an infinite loop during COBOL program execution, and the application does not terminate.

[Symptom 2]

Under Condition B, the runtime message JMP0601I-I/U CODE=111 (*) is output during the execution of a SORT statement or MERGE statement.

*: CODE=111 signifies an incorrect record length.

[Environment]

PowerBSORT (64bit) is not installed.

[Condition A]

1. Use NetCOBOL for Windows(64) V12.0.0 or later.
2. The record length of the sort-merge file meets one of the following conditions:
    - (2-1) The file records are fixed-length, and one of the following is true:
        - The record length is from 21481 bytes to 21484 bytes.
        - Compiler option EQUALS is specified, and the record length is from 21473 bytes to 21480 bytes.
    - (2-2) The file records are variable-length, and one of the following is true:
        - The record length is from 21477 bytes to 21484 bytes.
        - Compiler option EQUALS is specified, and the record length is from 21469 bytes to 21476 bytes.
3. Execute a SORT statement or a MERGE statement for the file specified in condition 2.

Note:

Compiler option EQUALS (processing method for identical key data in SORT statements):

At runtime, if multiple records with the same key exist during input to a SORT statement, this option guarantees that the order of the output records from the SORT statement matches the order of the input records. The default value is NOEQUALS (not guaranteed).

[Condition B]

1. Use NetCOBOL for Windows(64) V12.0.0 or later.
2. The record length of the sort-merge file meets one of the following conditions:
   - (2-1) The record length is from 21485 bytes to 32760 bytes.
   - (2-2) The file records are fixed-length, compiler option EQUALS is specified, and the record length is from 21481 bytes to 32760 bytes.
   - (2-3) The file records are variable-length, compiler option EQUALS is specified, and the record length is from 21477 bytes to 32760 bytes.
3. Execute a SORT statement or a MERGE statement for the file specified in condition 2.

PH22132

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, when executing a COBOL application, if an SQL statement is executed with a data length of 0 specified for a host variable of variable-length string type, the host variable data is not set as empty string data (character string data with a length of 0). Consequently, executing the SQL statement might set data other than an empty string in the table record, or the intended table might not be derived.

[Condition]

1. Use a Windows system.
2. Create the COBOL program as a Unicode application by specifying one of the following compiler options:
   - ENCODE(UTF8,UTF32,LE)
   - ENCODE(UTF8,UTF32,BE)
3. Access the database through the ODBC interface.
4. Set the length field of a national host variable that handles variable-length character string data to 0.
5. Execute an SQL statement with the host variable from condition 4 specified in one of the following locations:
   - An insert value within the insert value list of an INSERT statement.
   - A value specification within the SET clause of an UPDATE statement.
   - A search condition in a WHERE clause.
   - The USING clause of a dynamic SQL statement.
   - The argument list of a CALL statement.

PH22227

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, the runtime message JMP0086I-W is output during COBOL application execution, and execution of the SQL statement fails. The following information is set in SQLSTATE/SQLCODE/SQLMSG:

```
SQLSTATE=?????
SQLCODE =-999999992
SQLMSG = An invalid process occurred.
```

[Condition]

1. Use a Windows system.
2. Access the database through the ODBC interface.
3. Create the COBOL program as a Unicode application by specifying one of the following compiler options:
   - ENCODE(UTF8,UTF32,LE)

- ENCODE(UTF8,UTF32,BE)

4. Use a national host variable with a length exceeding 4085 characters and execute a SELECT statement or a FETCH statement to retrieve data (*).

*: The upper limit for the number of characters in a national host variable when specifying ENCODE(UTF8,UTF32,LE) or ENCODE(UTF8,UTF32,BE) is as follows:

- Fixed-length national item: 8173 characters
- Variable-length national item: 8172 characters

PH22522

- V/L: V11.0.0 to V12.2.0A

Under the following condition, when a COBOL application is executed, the control character carriage return (CR) in records read from a line sequential file is not replaced by spaces.

For variable-length files, the effective record length reported to the DEPENDING ON specified data item specified in the record clause is the length including the return of the control character.

Example: When reading "あ" (one full-width character) from a line sequential file with three-character national items defined in the FILE SECTION, encoded as UTF-32,LE, and using CR LF as the line feed character.

- Input records from a line sequential file

あ	Control character carriage return (CR)	Control character line feed (LF)
`42 30 00 00`	`0D 00 00 00`	`0A 00 00 00`

- Result of reading an input record

あ	Control character carriage return (CR)	Full-width space
`42 30 00 00`	`0D 00 00 00`	`00 03 00 00`

When the COBOL runtime reads a record shorter than the one defined in the FILE SECTION, it removes control characters and pads with spaces, but the carriage return (CR) control character remains in the record instead of being replaced with an full-width space.

- Expected read result

あ	Full-width space	Full-width space
`42 30 00 00`	`00 03 00 00`	`00 03 00 00`

[Condition]

1. Use a line sequential file.
2. Specify high-speed file processing (*1) for the file in condition 1.
3. Declare the record (*2) defined in the FILE SECTION of the file in condition 1 using only national items.
4. The national items in condition 3 meet one of the following conditions:
    a. No ENCODING clause is specified, and one of the following compiler options is in effect:
        - ENCODE(UTF8,UTF32)
        - ENCODE(UTF8,UTF32,LE)
        - ENCODE(UTF8,UTF32,BE)

   b. Specify, in the ENCODING clause of the FILE SECTION for the file in condition 1, an alphabet-name associated with one of the following encoding methods in the ALPHABET clause:

      - UTF32
      - UTF32BE
      - UTF32LE

5. Assign a text file meeting the following conditions to the file in condition 1:

   - Encoded in UTF-32
   - Uses carriage return line feed (CRLF) or carriage return (CR) control characters as line feed characters

6. Execute an OPEN statement with the INPUT phrase.
7. The data of the record read by the READ statement is shorter than the record length defined in condition 3.

*1: High-speed file processing can be specified using the following methods:

- When specifying a file-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when setting the environment variable.

```
file-name=[path-name]file-name,BSAM
```

- When specifying a data-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when specifying the data-name in the program.

```
MOVE "[path-name]file-name,BSAM" TO data-name
```

- When specifying a literal file-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when specifying the literal file-name in the program.

```
ASSIGN TO "[path-name]file-name,BSAM"
```

- For batch specification: Specify "BSAM" in the environment variable @CBR_FILE_SEQUENTIAL_ACCESS.

```
@CBR_FILE_SEQUENTIAL_ACCESS=BSAM
```

*2: Applies to both fixed-length and variable-length record formats.

PH22524

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, invalid characters are inserted into the data read from a file during application execution:

1. Use a line sequential file.
2. Specify high-speed file processing (*1) for the file in condition 1.
3. Declare the record (*2) defined in the FILE SECTION of the file in condition 1 using only national items.
4. The national items in condition 3 meet one of the following conditions:

   a. No ENCODING clause is specified, and one of the following compiler options is in effect:

      - ENCODE(UTF8,UTF32)
      - ENCODE(UTF8,UTF32,LE)
      - ENCODE(UTF8,UTF32,BE)

   b. An alphabet-name associated with one of the following encoding forms in the ALPHABET clause is specified in the ENCODING clause of the FILE SECTION for the file in condition 1:

      - UTF32
      - UTF32BE

        - UTF32LE

5. The text file assigned to the file in condition 1 is encoded in UTF-32.
6. Specify compiler option NSP (*3).
7. Execute an OPEN statement with the INPUT phrase.
8. The data of the record read by the READ statement is shorter than the record length defined in condition 3.

*1: High-speed file processing can be specified using the following methods:

- When specifying a file-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when setting the environment variable.

```
file-name=[path name]file name,BSAM
```

- When specifying a data-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when specifying the data-name in the program.

```
MOVE "[path name]file name,BSAM" TO data-name
```

- When specifying a literal file-name for the file reference in the program: Specify ",BSAM" after the file name of the file to be assigned when specifying the literal file-name in the program.

```
ASSIGN TO "[path name]file name,BSAM"
```

- For batch specification: Specify "BSAM" in the environment variable @CBR_FILE_SEQUENTIAL_ACCESS.

```
@CBR_FILE_SEQUENTIAL_ACCESS=BSAM
```

*2: Applies to both fixed-length and variable-length record formats.

*3: Compiler option NSP is specified by default.

PH22665

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, data truncation might occur when executing an SQL statement during COBOL application execution (*1).

If truncation occurs, the SQLSTATE/SQLCODE/SQLMSG values set by the ODBC driver are reported to the COBOL application (*2).

[Condition]

1. Use a Windows system.
2. Access the database through the ODBC interface.
3. Create the COBOL program as a Unicode application by specifying one of the following compiler options:
    - ENCODE(UTF8,UTF32,LE)
    - ENCODE(UTF8,UTF32,BE)
4. Execute an SQL statement using a national host variable containing 4-byte characters (surrogate pairs) (*3).

*1: When executing an SQL statement, if the data contains 4-byte characters (surrogate pairs), character splitting might occur in addition to truncation. Whether splitting occurs depends on the implementation of the ODBC driver being used. When using the Enterprise Postgres ODBC driver, splitting might occur when retrieving data.

*2: Because truncation occurs on the database side, the behavior when truncation occurs follows the database's behavior. If truncation occurs when updating database data by executing an INSERT statement, an UPDATE statement, or similar, execution of the SQL statement fails. The truncated data is not stored in the database. If truncation occurs when retrieving database data using a SELECT statement, a FETCH statement, or similar, the data after truncation is retrieved.

*3: If data truncation occurs when referencing 4-byte characters (surrogate pairs), the SQLSTATE/SQLCODE/SQLMSG values returned by the ODBC driver are reported to the COBOL program. If character splitting occurs, the runtime system outputs JMP0086I-W and sets the following SQLSTATE/SQLCODE/SQLMSG values:

```
SQLSTATE=?????
SQLCODE =-999999992
SQLMSG = An illegal process occurred.
```

PH22812

- V/L: V12.2.0 to V12.2.0A

Under the following conditions, either Symptom 1 or Symptom 2 occurs:

[Symptom 1]

When a COBOL application performs data communication with another application process via a pipe, the ACCEPT statement might not receive data correctly.

[Symptom 2]

When a COBOL application performs data communication with another application process via a pipe, the following runtime error might occur in the ACCEPT statement: JMP0208I-E INVALID NUMERICAL VALUE ACCEPTED.

[Condition]

1. The COBOL source program includes an ACCEPT statement with one of the following specifications:
    - No mnemonic-name specified (no mnemonic-name specified for a function-name).
    - A mnemonic-name for the function-name SYSIN.
    - A mnemonic-name for the function-name CONSOLE.

    Symptom 1: The input item of the ACCEPT statement is an alphanumeric item.

    Symptom 2: The input item of the ACCEPT statement is a numeric item.
2. Execute a COBOL program that meets one of the following conditions:
    - The COBOL source program is compiled with compiler option MAIN(MAIN) specified.
    - The environment variable information @CBR_CONSOLE=SYSTEM is set during COBOL program execution.
3. Perform data communication between another application process and the COBOL application process via a pipe (*).

*: The problem has been confirmed to occur in the following cases:

- When performing data communication using iii/Win and the ACCEPT statement.
- When executing using the "Run" function in NetCOBOL Studio.

PH23281

- V/L: V11.0.0 to V12.2.0A

Under the following conditions, trailing blanks in records written to a line sequential file are not removed during COBOL application execution:

1. Specify REMOVE for the environment variable @CBR_TRAILING_BLANK_RECORD (*1).
2. Use a line sequential file.
3. Declare the record (*2) defined in the FILE SECTION of the file in condition 2 using only national items.
4. The national items in condition 3 meet one of the following conditions:
    a. No ENCODING clause is specified, and one of the following compiler options is in effect:
        - ENCODE(UTF8,UTF32)
        - ENCODE(UTF8,UTF32,LE)
        - ENCODE(UTF8,UTF32,BE)

   b. An alphabet-name associated with one of the following encoding forms in the ALPHABET clause is specified in the ENCODING clause of the FILE SECTION for the file in condition 2:

      - UTF32
      - UTF32BE
      - UTF32LE

5. The text file assigned to the file in condition 2 is encoded in UTF-32.
6. Specify compiler option NSP (*3).
7. Execute an OPEN statement with either the OUTPUT or EXTEND phrase.
8. Write data shorter than the record length defined in condition 3 using a WRITE statement..

*1: The default value of the environment variable @CBR_TRAILING_BLANK_RECORD is VALID (trailing blanks in records are treated as valid data).

*2: Applies to both fixed-length and variable-length record formats.

*3: The default value of compiler option NSP is NSP (spaces in national items are treated as national spaces).

PH24472

- V/L: V10.2.0 to V12.2.0A

Under the following conditions, during COBOL program execution, runtime message JMP0259I-W (*1) and the output string specified in the DISPLAY statement are not output to the event log for the computer. The program then terminates with runtime message JMP0259I-U.

[Condition]

1. The DISPLAY statement specifies the function-name SYSOUT (*2), SYSERR, or CONSOLE, or a user-defined name associated with one of these function-names.
2. An environment variable (*3) is specified that redirects output from each function-name to the event log.
3. The output string specified in the DISPLAY statement cannot be written to the event log of the computer whose name is specified in square brackets in the environment variable in condition 2 (*4).

*1: Runtime messages JMP0259I-W/U have the following format:

```
DISPLAY OUTPUT FOR EVENTLOG CANNOT BE WRITTEN TO COMPUTER '$1'. '$2' $3
```

$1 represents the computer name specified as the event log output destination. $2 represents the output string specified in the DISPLAY statement. $3 represents the error address.

If the severity code is W, this message is written to the event log of the local computer, and processing continues. If the severity code is U, the program terminates abnormally.

*2: The following are equivalent to specifying the function-name SYSOUT:

- Specifying the function-name SYSPUNCH
- Specifying a user-defined name associated with the function-name SYSPUNCH
- Omitting a function-name

*3: The relevant environment variables are:

- @CBR_DISPLAY_SYSOUT_OUTPUT=EVENTLOG[(computer-name)]
- @CBR_DISPLAY_SYSERR_OUTPUT=EVENTLOG[(computer-name)]
- @CBR_DISPLAY_CONSOLE_OUTPUT=EVENTLOG[(computer-name)]

*4: This includes, for example:

- The computer-name within the square brackets in the environment variable in condition 2 does not exist on the network.

- The computer-name within the square brackets in the environment variable in condition 2 is valid, but writing to the event log of that computer is not possible.

PH24511

- V/L: V10.1.0 to V12.2.0A

Under the following environment and conditions, runtime message JMP0320I-I/U (FDBK=7) is output during COBOL application execution. FILE STATUS is set to "90".

[Environment]

Actian Zen and Actian DataExchange are installed.

[Condition]

1. Use a Btrieve file (*1).
2. Replicate the file in condition 1 using the data replication function of Actian DataExchange (*2).
3. The file in condition 1 is an indexed file with a secondary key.
4. The access mode (*3) of the file in condition 1 is SEQUENTIAL or DYNAMIC.
5. Execute an OPEN statement with the I-O phrase for the file in condition 1.
6. Perform one of the following operations on the file in condition 1:
   - (6-1) When the access mode in condition 4 is SEQUENTIAL:
     1. Execute a START statement to position the file using the secondary key.
     2. Execute a READ statement.
     3. Execute a REWRITE statement.
     4. Execute a READ statement.
   - (6-2) When the access mode in condition 4 is DYNAMIC:
     1. Execute a START statement to position the file using the secondary key.
     2. Execute a READ statement with the NEXT phrase.
     3. Execute a REWRITE statement.
     4. Execute a READ statement with the NEXT phrase.

*1: A Btrieve file can be specified using the following methods:

- When a literal file-name is specified in the ASSIGN clause: Specify ",BTRV" after the file name of the file to be assigned in the program.

```
ASSIGN TO "[path name]file name,BTRV"
```

- When a data-name is specified in the ASSIGN clause: Specify ",BTRV" after the file name of the file to be assigned in the program.

```
MOVE "[path name]file name,BTRV" TO data-name
```

- Environment variable format: Specify ",BTRV" after the file name of the file to be assigned when setting the environment variable.

```
file-name=[path name]file name,BTRV
```

*2: For details on the data replication function of Actian DataExchange, refer to the Actian DataExchange documentation.

*3: The access mode is specified using the ACCESS MODE clause.

- SEQUENTIAL specifies sequential access.
- DYNAMIC specifies dynamic access.
- Dynamic access allows both sequential and random processing.

3.3 J Adapter Class Generator

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

There is no information on the program correction.

3.4 PowerFORM Runtime

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

V/L indicates the range of versions where the defect existed.

PH15160

- V/L: V11.0.0 to V12.2.0A

Under the following condition, error might occur.

[Condition]

1. Block text field, block UCS2 field, or a block mixed text field is set to the form.
2. The widths are smaller than the width of the font size and the width of the font pitch.
3. One of the following is specified:
    - Forbidden at start-of-line
    - Forbidden at end-of-line
    - Force punctuation at line-ends

3.5 Fujitsu Mainframe Floating-Point Arithmetic Emulator

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows

There is no information on the program correction.

3.6 PowerBSORT

Information described here applies to the following products.

- NetCOBOL Enterprise Edition Developer (64bit) for Windows
- NetCOBOL Enterprise Edition Server Runtime (64bit) for Windows

V/L indicates the range of versions where the defect existed.

PH15958

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, the following error might occur in the BSORT function, causing PowerBSORT execution to fail:

- Error detail code (*1): 212 (BSERR_TEMPSPACE)

If message output is specified (*2), the error message "Insufficient temporary file space" is also output.

*1: errdetail member of the BSRTREC structure

*2: msglevel member and optionfunc member of the BSRTPRIM structure, or msgfile_addr member of the BSRTFILE structure

[Condition]

1. Use the BSORT function from a C language program to utilize PowerBSORT.
2. Specify the sort function (BS_SORT) in the function member of the BSRTPRIM structure.
3. Specify, in the tmpfile_tbl member of the BSRTFILE structure, the addresses of two or more BSFILE structures that specify temporary file path names.
4. One temporary file runs out of space, while other temporary files have free space (*3).

*3: A similar issue occurs if all temporary files run out of space; however, this is expected behavior.

PH16294

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, two records with identical key field values might be output during PowerBSORT execution.

[Condition]

1. Use the sort or merge function.
2. Specify fixed fields for the text file.
3. Use the suppression option (*1) (*2).
4. Use the record reconstruction option (*3) (*4).
5. The record length, excluding the newline character, resulting from the record reconstruction option in condition 4 is as follows:
    - For 64bit products: Not a multiple of 8
6. Do not specify a key field (*5).

*1: This option retains one record and removes others when the key field values are identical.

*2: Excluding cases where the first or last operand is specified for the record aggregation option (-summary) of the bsortex command.

*3: Excluding cases where the position from the specified position in the input record to the end is specified in the reorganization field (using the pos.END format).

*4: When the record reconstruction option (reconst operand) is specified for the input file information option (-input) of the bsortex command.

*5: If no key field is specified, the entire record (in this case, the record reorganized by the record reconstruction option) is considered the key field.

PH17490

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, PowerBSORT might terminate abnormally with a non-zero exit code.

Examples of abnormal termination:

- Message displayed in Windows: "PowerBSORT bsortex application has stopped working"
- Exception codes recorded in the Windows Event Log: 0xc0000005, 0xc0000374

[Condition]

1. Use the bsortex command.

2. Use one or both of the following functions:
    - Record reconstruction option for the output file
    - Specification of a record separator character in the output file
3. The output record length resulting from the functions in condition 2 meets either of the following conditions:
    - It is longer than the input record length.
    - It is longer than the reconstructed record length specified by the record reconstruction option for the input file.

Notes:

The functions in the bsortex command are specified using the following options and operands:

- Record reconstruction option for the output file: -output option with the reconst operand
- Input record length: -input option with the reclen operand
- Record reorganization for the input file: -input option with the reconst operand

PH24587

- V/L: V10.1.0 to V12.2.0A

Under the following conditions, the command might terminate with an error and output a message (*1), even if the command options are specified correctly.

*1: One of the following messages is output:

- 0130 Invalid output file system is specified
- 0144 Output file path name is not specified.

[Condition]

1. Use the bsort command.
2. When the I/O file system option (-F) (*2) is used, and a file system other than "btrv" (the Btrieve file system) (*3) is specified for the input file.

*2: The input/output file system option is specified as follows:

```
-F ofs,ifs [/ownername] [,ifs [/ownername] ...]
```

- ofs: Specifies the file system for the output file.
- ifs: Specifies the file system for the input file.
- ownername: Specifies the owner name (up to 8 characters) if required by the Btrieve file system.

*3: This includes cases where the file system specification for the input file is omitted.

Chapter 4 Notes to Consider When a Japanese Native Application is Converted to Run Globally

This chapter describes what is involved in taking a Japanese native application global.

4.1 Assumption

The program resources and run time code-set are made by Unicode.

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

4.2 Environments

There are following notes in the environments.

- This product must only be installed in a folder which has the only ASCII character set.
- When cobmkmf is used, the character outside ASCII cannot be used for the file name and the folder name of program resources.

4.3 Languages

The language used by this product (GUI, messages, etc.) is determined according to the setting of the display language of the execution environment of the Windows system.

By default, if the set display language cannot be used, English is selected as the default.

The default of the display language of the user account can be set depending on the following criteria.

- "Language" of Control Panel

Compilation messages

In this compiler, in accordance with the UI language during compilation, the COBOL command messages are set to Japanese or English, accordingly.

Runtime messages

The language of the Runtime Error Messages is decided when the runtime environment is established.

If there is no message of an applicable language, a Runtime Error Message is displayed in English.

NetCOBOL Studio

- When you use NetCOBOL Studio, define the applicable language of OS when the NetCOBOL product is installed, the system locale (This is set on the "Administrative" of "Region and Language" dialog box of the Control Panel), the display language (This is set on the "display language" of the Control Panel), and the language of the Format (This is set on the "Formats" of "Region and Language" dialog box of the Control Panel).
- Match the language of the client to the language of the server when you use the remote development function of NetCOBOL Studio. When the language of the client and the server is different, the message of the server might not be able to be displayed with NetCOBOL Studio of the client.

PowerFORM

When you use PowerFORM, define the same OS language for the system locale (This is set on the "Administrative" of "Region and Language" dialog box of the Control Panel) and the display language (This is set on the "display language" of the Control Panel) when the NetCOBOL product is installed.

4.4 Feature Difference with Japanese Version

In NetCOBOL, there are a Japanese version and a Global version. This product is a Global version.

This section explains the feature differences between a Japanese version and a Global version.

4.4.1 Related Products

The following NetCOBOL family products are not supported in Global version.

- FORM
- FORM overlay option

The following component is not supported in Global version:

- MeFt/Web HTML conversion functionality

The following Fujitsu products are not supported in Global version:

- Interstage Application Server
- Interstage Business Application Server
- Interstage Charset Manager
- Interstage List Creator
- Interstage List Works
- PowerRDBconnector
- SIMPLIA TF-MDPORT Pro
- SIMPLIA TF-LINDA
- SIMPLIA TF-EXCOUNTER
- SIMPLIA MF-STEPCOUNTER
- Symfoware Server

Note

For the following products, the name is different with the Japanese version.

Global version	Japanese version
PowerFORM RTS	MeFt
PowerBSORT	PowerSORT

4.4.2 Specifications

The following specifications are different with the Japanese version.

Currency symbol

In this compiler, currency sign of OS is used as Currency Symbol character.

If character other than the currency sign of OS is used, the method of specifying may differ based on the character intended to use.

- Characters with same code as $(X'24') or \(X'5C')

  Specified with/based on compile option CURRENCY.

  For details, refer to "CURRENCY(currency symbol handling)" in "NetCOBOL User's Guide."

- 1 byte character other than this

  CURRENCY SIGN clause is used, specified inside the source program.

  For details, refer to "CURRENCY SIGN Clause" in "NetCOBOL Language Reference."

Note

- Characters consisting of multiple bytes cannot be used as currency symbol character. For possible characters that can be used as currency symbol, refer to "CURRENCY SIGN Clause" in "NetCOBOL Language Reference."
- If currency sign of OS cannot be used as currency symbol, in that case \ (X'5C') will be taken as currency symbol.
- If data is passed between compilation units with different currency symbols, results might not be as expected. Specify the compilation option CURRENCY or CURRENCY SIGN clause to use the same currency symbol.

Handling of National item spaces

In this compiler, national item space (Trailing Blank and Figurative Constant SPACE) of Unicode encoding is handled as alphabetic spaces (U+0020).

Specify compiler option NSP, if national spaces are to be changed.

For details, refer to "NSP(Handling of spaces related to national item)" in "NetCOBOL User's Guide."

Note

If data is passed between compilation units with different National item spaces, results might not be as expected. Specify the compiler option NSP, and use the same national item space.

Printing Function

PDF Output:

The Japanese version supports PDF output, but the Global version does not.

Paper Size:

The default paper size for the Japanese version is A4. For the Global version, it is LETTER.

To change the paper size, specify it in the print information file or an I control record.

Printing font

If you use a print file without a FORMAT clause, a default font name according to the setting of the display language of the execution environment of the Windows system will be used.

When the display language is Japanese, the default font name is "MS Mincho, MS Gothic" and with any other language, the default font name is "Courier New".

In this scenario, the printing font value can be changed by the environment variables.

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

Output Characters:

The Japanese version supports Unicode and Shift-JIS character ranges. The Global version supports Unicode and ASCII.

Form Descriptors:

The Japanese version supports SMD and PMD. The Global version supports only PMD.

Electronic Form Output:

The Japanese version supports electronic form output, but the Global version does not.

4.4.3 PowerFORM

- The following function is not supported in Global version.
    - Making of Form Descriptors (PXD) for Class Interface.
    - Selection of Target System of the overlay pattern table. Target System becomes Windows fixed.
    - Overlay conversion of Template Image.
- When PowerFORM is used, the only ASCII characters can be used.
- When Form Descriptors is edited, the initial values of the size of the form and the font-name, etc. are different.

4.4.4 PowerFORM Runtime

When the Form is output in a Global version, it is necessary to specify the following.

- Specify Unicode for an application code.

  For details, refer to "How to Use PowerFORM RTS" in "PowerFORM Runtime Reference."
- Specify the font name by an English name.

  For details, Refer to " Printer Information File" in "PowerFORM Runtime Reference."
    - MAPFONT (Map font names)
- Specify the printer information file to which BOM(UTF-8) is added.

  For details, refer to "Printer Information File" in "PowerFORM Runtime Reference."
- Specify "USECHARTYPE UNI" with the printer information file. Adjust each character-code by "UNICODEN" and "UNICODEW" when the output result is not correct.

  For details. refer to "Printer Information File" in "PowerFORM Runtime Reference."
    - USECHARTYPE (Unicode full-width/half-width character table)
    - UNICODEW (Unicode full-width character range)
    - UNICODEN (Unicode half-width character range)

- The default value of the font name is different according to the locale.

  For details. refer to "Printer Information File" in "PowerFORM Runtime Reference."
    - MAPFONT (Map font names)