New Era of Networks Rules and Formatter Support for WebSphere® MQ Integrator

COBOL User Exits Supplement

Version 5.6

New Era of Networks Rules and Formatter Support for WebSphere® MQ Integrator

COBOL User Exits Supplement

Version 5.6

Note: **Before using this information, and the product it supports, be sure to read the general information under** *Notices* on page 33**.**

First Edition (December 2001)
This edition applies to New Era of Networks Rules and Formatter Support, Version 5.6, for IBM ® WebSphere MQ Integrator and to all subsequent releases and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page titled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories
Information Development,
Mail Point 095,
Hursley Park,
Winchester,
Hampshire,
England,
SO21 2JN.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1
Introduction

On z/OS, you can customize New Era of Networks Formatter by creating user exits using COBOL.

For example, you can create a user exit when the standard reformatting types of New Era of Networks Formatter do not meet your data needs. Use the user exit APIs provided with New Era of Networks Formatter to create your own user exits.

COBOL user exits function in the following manner:

- New Era of Networks Formatter calls a routine named NNGetUserExitFunctionPtrs () to provide the function address. The user-defined exit code shares a location with New Era of Networks Formatter.
- New Era of Networks Formatter takes a field from a parsed format and passes the field to the user exit.
- The value changes as part of the Reformat() function, and the new value is passed back to the field.

For more information, see the ***New Era of Networks Formatter Programming Reference***.

About This Document

The ***COBOL User Exit Supplement*** is designed for those who are responsible for New Era of Networks Rules and Formatter component administration. The system administrator should have an overall understanding of the Rules and Formatter Support for WebSphere MQ Integrator product and how it works. It is assumed that the system administrator is responsible for Rules

and Formatter Support for WebSphere MQ Integrator setup, configuration, and testing. The system administrator should be supported by a database administrator, who administers the databases interacting with Rules and Formatter Support for WebSphere MQ Integrator, and a network administrator, who ensures that network communications are set up to work with Rules and Formatter Support for WebSphere MQ Integrator.

The guide is organized into the following chapters:

- *Chapter 1: Introduction* provides an outline of the contents of this supplement and the documentation set.
- *Chapter 2: COBOL User Exits* describes how to develop COBOL user exits to customize New Era of Networks Formatter.
- *Appendix A: Sample Programs* provides sample JCL and user exit programs.
- *Appendix B: Notices* provides IBM trademark and service mark information.

Documentation Set

The Rules and Formatter Support for WebSphere MQ Integrator documentation set includes:

- ***User's Guide***
- ***System Management Guide***
- ***New Era of Networks Formatter Programming Reference***
- ***New Era of Networks Rules Programming Reference***
- ***Application Development Guide***
- *COBOL User Exits Supplement*

Document Conventions

The following document conventions are used in this guide.

Text	Convention	Example
code	courier	`<user ID> <password>`
command line display	courier	`The message successfully parsed.`
command line entry	courier bold	**`NNFAD-t`**
command line prompt	courier	`Enter the input file name:`
path	regular	ora/bin (UNIX) ora\bin (NT)
book names	bold, italic	***User's Guide***
chapter and section names	italic	*NT Installation*

Chapter 2

COBOL User Exits

By creating COBOL user exits on z/OS, you can externally customize New Era of Networks Formatter to meet your own data requirements. For example, you can:

- Enhance data received from another database source before sending it to its final destination.
- Manipulate data in a manner that is not currently supported by New Era of Networks Formatter, such as adding field values within a repeating structure.
- Call Rules and Formatter Support for WebSphere MQ Integrator functions by statically linking the user exit APIs to the user application.

Creating User Exits

When a user exit is invoked through a format, New Era of Networks Formatter executes C++ user exits first. If a C++ user program cannot be found, New Era of Networks Formatter calls the COBOL program and executes the COBOL user exits.

When combining COBOL and C++ programs to create user exits, the following occurs:

- COBOL statically calls C functions
- Only one C/C++ load module is linked to a COBOL program
- C/C++ dynamically calls C++ functions

- COBOL user exits must be compiled and linked as a DLL.

Note:
For more information on creating user exits in C++, see ***New Era of Networks Formatter Programming Reference*** and the ***System Management Guide***.

Naming Conventions

COBOL user exits exist as standalone COBOL programs. The name of a user exit must be equal to the name of the COBOL program in which it is implemented.

To create COBOL user exits:

1. Write the user exit program in COBOL and define the LINKAGE SECTION of the user exit program using the following sample:

   LINKAGE SECTION

   All parameters from New Era of Networks Formatter are passed by reference to the user application through the LINKAGE SECTION.

```
LINKAGE SECTION.
    01  SESSION-POINTER         USAGE POINTER.
    01  PARSED-FIELDS-POINTER    USAGE POINTER.
    01  OUTPUT-FIELD-LENGTH      PIC  S9(9)  BINARY.
    01  OUTPUT-FIELD             PIC  X(2000).
    01  RETURN-CODE-VALUE        PIC  S9(9)  BINARY.

PROCEDURE DIVISION USING BY REFERENCE SESSION-POINTER
                         PARSED-FIELDS-POINTER
                         OUTPUT-FIELD-LENGTH
                         OUTPUT-FIELD
                         RETURN-CODE-VALUE.
```

Parameter	Input/ Output	Description
SESSION-POINTER	Input	Sends a pointer to the DBMS session.

Parameter	Input/ Output	Description
PARSED-FIELDS-POINTER	Input	Sends the data to be transformed to the user exit
OUTPUT-FIELD-LENGTH	Output	Returns the length of the output field passed back to New Era of Networks Formatter.
OUTPUT-FIELD	Output	Returns the data that was transformed by the user exit code to New Era of Networks Formatter.
RETURN-CODE-VALUE	Output	Returns one of the following values to New Era of Networks Formatter. 0 = successful 16 = failed

2. Compile and link the user exit code:

   For sample JCL, see *Sample Programs* on page 19.

API Reference for COBOL Wrappers

COBOL user exit APIs allow the user application to access New Era of Networks Rules and Formatter from a COBOL application. This section details the COBOL wrapped user exit APIs.

NNParsedField

The following is a list of the COBOL wrapped APIs for the NNParsedField class. For more information, see ***New Era of Networks Formatter Programming Reference.***

GETFLDASC

GetFieldAscii() returns the ASCII value and length of the specified input parsed field.

Binary data is returned as a string preceded by 0x. For example, the binary value 12345 is returned as 0x00003039.

The INPUT-FIELD-NAME for all of the user exits should be passed to the exit routine as NULL Terminated Strings. For more information, see the IBM manual ***COBOL for OZ/390 & VM Programming Guide***.

Working Storage

```
01  PARSED-FIELDS                  POINTER.
01  INPUT-FIELD-NAME               PIC X(12)VALUE
                                    Z'MyFieldName'.
01  INSTANCE                       PIC S9(09) COMP.
01  INPUT-FIELD-VALUE.
    05  INPUT-FIELD-VALUE-BYTE     PIC X(01)
                                    OCCURS  2000 TIMES.
01  INPUT-FIELD-VALUE-LENGTH       PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTFLDASC' USING PARSED-FIELDS
                      INPUT-FIELD-NAME
                      INSTANCE
                      INPUT-FIELD-VALUE
                      INPUT-FIELD-VALUE-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-NAME	Input	Sends the name of input parsed field for which to return a character value.

Parameter	Input/ Output	Description
INSTANCE	Input	Sends a number indicating which instance (zero-based index) of input parsed field to return value for, in cases where a field name is used more than once in one or more input formats to construct the output message.
INPUT-FIELD-VALUE	Output	Returns a value of the input parsed field in character format.
INPUT-FIELD-NAME-LENGTH	Output	Returns the length, in bytes, of the value of the input parsed field.

GTINFLNM

GetCurrInFldName() returns the name and the length of the current input parsed field.

Working Storage

```
01  PARSED-FIELDS                POINTER.
01  INPUT-FIELD-NAME             PIC X(12)VALUE
                                  Z'MyFieldName'.
01  INPUT-FIELD-NAME-LENGTH      PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTINFLNM'  USING PARSED-FIELDS
                       INPUT-FIELD-NAME
                       INPUT-FIELD-NAME-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-NAME	Output	Returns a name of input parsed field.
INPUT-FIELD-NAME-LENGTH	Output	Returns the length of the input field name.

GTOTFLNM

GetCurrOutFldName() returns the name of the output field associated with the current input parsed field.

Working Storage

```
01  PARSED-FIELDS                POINTER.
01  OUTPUT-FIELD-NAME            PIC X(33).
01  OUTPUT-FIELD-NAME-LENGTH     PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTOTFLNM'  USING PARSED-FIELDS
                       OUTPUT-FIELD-NAME
                       OUTPUT-FIELD-NAME-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
OUTPUT-FIELD-NAME	Output	Returns the name of output parsed field.
OUTPUT-FIELD-NAME-LENGTH	Output	Returns a binary integer indicating the length of the output parsed field name.

GTINFLDT

GetCurrInFldData() and GetCurrInFldLength() return the raw data value of the current input parsed field.

Working Storage

```
01  PARSED-FIELDS                POINTER.
01  INPUT-FIELD-DATA.
    05  INPUT-FIELD-DATA-BYTE    PIC X(01)
                                  OCCURS  2000 TIMES.
01  INPUT-FIELD-DATA-LENGTH      PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTINFLDT'  USING PARSED-FIELDS
                       INPUT-FIELD-DATA
                       INPUT-FIELD-DATA-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-DATA	Output	Returns the data value of current input field.
INPUT-FIELD-DATA-LENGTH	Output	Returns a binary integer indicating the length of the input field data.

GTINFLST

GetCurrInFldAsciiData() returns the character value and length of the current input parsed field.

Binary data is returned as a string preceded by 0x. For example, the binary value 12345 is returned as 0x00003039.

Working Storage

```
01  PARSED-FIELDS                  POINTER.
01  INPUT-FIELD-DATA.
    05  INPUT-FIELD-DATA-BYTE      PIC X(01)
                                   OCCURS  2000 TIMES.
01  INPUT-FIELD-DATA-LENGTH        PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTINFLST'  USING PARSED-FIELDS
                       INPUT-FIELD-DATA
                       INPUT-FIELD-DATA-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-DATA	Output	Returns the data value of current input field.
INPUT-FIELD-DATA-LENGTH	Output	Returns a binary integer indicating the length of the input field data.

GTINFLLN

GetCurrInFldLength() returns the length of the current input parsed field.

Working Storage

```
01  PARSED-FIELDS                POINTER.
01  INPUT-FIELD-LENGTH           PIC  S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTINFLLN'  USING PARSED-FIELDS
                       INPUT-FIELD-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-LENGTH	Output	Returns a binary integer indicating length, in bytes, of the input field data.

GTINFLTP

GetCurrInFldType() returns the data type of the current input parsed field.

Working Storage

```
01  PARSED-FIELDS       POINTER.
01  INPUT-FIELD-TYPE    PIC S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTINFLTP'  USING PARSED-FIELDS
                       INPUT-FIELD-TYPE
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
INPUT-FIELD-TYPE	Output	Returns a data type of the current input parsed field. 0 = Not Applicable. 1 = String 3 = Binary 5 = Packed Integer 6 = Signed Packed Integer 7 = Zoned Integer 8 = Signed Zoned Integer

GTOTFLDT

GetCurrOutFldData() and Get CurrOutFldLength return the raw data value of the current output parsed field.

Working Storage

```
01  PARSED-FIELDS                POINTER.
01  OUTPUT-FIELD-DATA.
    05  OUTPUT-FIELD-DATA-BYTE   PIC X(01)
                                 OCCURS  2000 TIMES.
01  OUTPUT-FIELD-DATA-LENGTH     PIC   S9(09) COMP.
```

COBOL Calling Portion

```
CALL 'GTOTFLDT'  USING PARSED-FIELDS
                       OUTPUT-FIELD-DATA
                       OUTPUT-FIELD-DATA-LENGTH.
```

Parameter	Input/ Output	Description
PARSED-FIELDS	Input	Sends a pointer to the class that represents all parsed field values.
OUTPUT-FIELD-DATA	Output	Returns the data value of current output field.
OUTPUT-FIELD-DATA-LENGTH	Output	Returns a binary integer indicating the length of the output field data.

Sample COBOL User Exit Program

TESTCOB1 is a sample COBOL user exit application. The program calls all COBOL user exit APIs, combines the output of each API into a field, and passes the field back to New Era of Networks Formatter as the output field value.

In order to use the sample exit, you must create a user exit named TESTCOB1 using the New Era of Networks Formatter GUI and specify an Exit Routine that is also identified as TESTCOB1. You must then apply an output control that uses the exit to reformat your parsed messages. When the control is invoked, New Era of Networks Formatter calls the TESTCOB1 program which executes using the parsed fields from your message as input.

For the complete sample, see *Sample Programs* on page 19.

Appendix A

Sample Programs

This appendix contains the following sample JCL and user exit programs:

- *Sample JCL for Compiling COBOL User Exit Programs*
- *Sample JCL for Compiling COBOL User Exit Programs that Access DB2*
- *Sample Program for COBOL User Exits*
- *Sample Program for COBOL User Exits Using DB2*
- *Sample Program for C++ User Exits Using DB2*

Sample JCL for Compiling COBOL User Exit Programs

```
// (Put JOBCARD here)//*
//PROCS    JCLLIB ORDER=IGY.V2R2M0.SIGYPROC
//*
//COBOL    EXEC IGYWCL,
//         LNGPRFX=IGY.V2R2M0,
//         LIBPRFX=CEE,
//         PARM.COBOL='DLL,EXPORTALL,RENT,PGMNAME(LONGMIXED)',
//
PARM.LKED='(RENT,LIST,XREF,LET,MAP,DYNAM(DLL),CASE(MIXED),
//             COMPAT(CURRENT))'
//*
//COBOL.SYSIN DD DSN=YOUR.SOURCE.PDS(URPROG),DISP=SHR
//*
//LKED.SYSLMOD DD PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='/(LIBPATH Directory)/URPROG'
```

```
//LKED.SYSDEFSD DD PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='/(AnyDirectory)/URPROG.x'
//LKED.EXITWRAP DD PATHOPTS=(ORDONLY),
//           PATH='/(Directory containing nnt56exitwrpr.x)
nnt56exitwrpr.x'
//LKED.SYSIN  DD *
  INCLUDE EXITWRAP
```

Sample JCL for Compiling COBOL User Exit Programs that Access DB2

```
// (Put JOBCARD here)
//*
//PROCS    JCLLIB ORDER=IGY.V2R2M0.SIGYPROC
//*
//PC      EXEC
PGM=DSNHPC,PARM='HOST(COBOL),ATTACH(RRSAF)',
//             REGION=4096K
//DBRMLIB DD DSN=YOUR.DBRMLIB(URDBRM),
//         DISP=SHR
//STEPLIB  DD  DISP=SHR,DSN=DSN610.SDSNEXIT
//         DD  DISP=SHR,DSN=DSN610.SDSNLOAD
//SYSCIN   DD  DSN=&&DSNHOUT,DISP=(MOD,PASS),UNIT=SYSDA,
//             SPACE=(800,(500, 500))
//SYSLIB DD DSN=YOUR.DCLGEN.LIB,
//          DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSTERM  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSUT1   DD  SPACE=(800,(500, 500),,,ROUND),UNIT=SYSDA
//SYSUT2   DD  SPACE=(800,(500, 500),,,ROUND),UNIT=SYSDA
//COBOL    EXEC IGYWCL,
//         LNGPRFX=IGY.V2R2M0,
//         LIBPRFX=CEE,
//
PARM.COBOL='DLL,EXPORTALL,RENT,PGMNAME(LONGMIXED)',
```

```
//
PARM.LKED='(RENT,LIST,XREF,LET,MAP,DYNAM(DLL),CASE(MIXE
),
//             COMPAT(CURRENT))'
//*
//COBOL.SYSIN DD DSN=&&DSNHOUT,DISP=SHR
//*
//LKED.SYSLMOD DD PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='/(LIBPATH Directory)/URPROG'
//LKED.SYSDEFSD DD PATHOPTS=(OWRONLY,OCREAT),
//            PATHMODE=(SIRWXU,SIRWXG),
//            PATH='/(AnyDirectory)/URPROG.x'
//LKED.EXITWRAP DD PATHOPTS=(ORDONLY),
//           PATH='/(Directory containing
nnt56exitwrpr.x)/nnt56exitwrpr.x'
//LKED.SYSIN  DD *
  INCLUDE EXITWRAP
  INCLUDE DB2LIB(DSNRLI)
```

Sample Program for COBOL User Exits

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. "TESTCOB1".
       AUTHOR. - NEW ERA OF NETWORKS.

       ENVIRONMENT DIVISION.
       CONFIGURATION SECTION.
       *SOURCE-COMPUTER. IBM-370 WITH DEBUGGING MODE.

       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01 WS-FIELD-DATA        PIC X(32000) VALUE SPACES.
       01 WS-FLD-NAME          PIC X(33)    VALUE SPACES.
       01 WS-FIELD-NAME        PIC X(33)    VALUE SPACES.
```

```
01 WS-FIELD-LENGTH      PIC S9(09)   COMP VALUE 0.
01 WS-INDEX             PIC S9(09)   COMP VALUE 0.
01 WS-FIELD-TYPE        PIC S9(09)   COMP VALUE 0.
01 WS-FLD-TYPE          PIC 9(09)    VALUE 0.
01 WS-MAX-DATA-LENGTH   PIC S9(09)   COMP VALUE +32000.
01 WS-DATA-LENGTH       PIC S9(09)   COMP VALUE 0.
01 WS-ARRAY-SUB         PIC S9(09)   COMP VALUE 0.
01 WS-OUTPUT-LEN        PIC S9(09)   COMP.

01 WS-ARRAY.
   05 WS-ARRAY-BYTE     PIC X
                        OCCURS 32000 TIMES.

01 WS-OUTPUT-LEN        PIC S9(09)   COMP.
01 WS-OUTPUT-FIELD.
   05 WS-OUTPUT-BYTE    PIC X
                        OCCURS 32000 TIMES
                        INDEXED BY OUTPUT-IDX.
01 WS-RET-CODE          PIC S9(09)   COMP.

LINKAGE SECTION.
01 SESSION-POINTER      POINTER.
01 PARSED-FIELDS        POINTER.
01 OUTPUT-LENGTH        PIC S9(09)   COMP.
01 OUTPUT-FIELD         PIC X(32000).
01 RET-CODE             PIC S9(09)   COMP.

PROCEDURE DIVISION USING SESSION-POINTER
                         PARSED-FIELDS
                         OUTPUT-LENGTH
                         OUTPUT-FIELD
                         RET-CODE.

MOVE WS-MAX-DATA-LENGTH           TO WS-FIELD-LENGTH.

CALL 'GTOTFLDT' USING PARSED-FIELDS
                      WS-FIELD-DATA
                      WS-FIELD-LENGTH
     ON EXCEPTION
        DISPLAY 'ERROR ON GTOTFLDT CALL'
END-CALL.
```

```
DISPLAY ' OUTPUT FIELD ON ENTRY IS: '
       WS-FIELD-DATA (1:WS-FIELD-LENGTH).
DISPLAY 'OUTPUT LENGTH ON ENTRY IS: ' WS-FIELD-LENGTH.

MOVE SPACES                          TO WS-OUTPUT-FIELD
                                        WS-FIELD-DATA.
MOVE +0                              TO WS-OUTPUT-LEN
                                        WS-FIELD-LENGTH.
SET OUTPUT-IDX                       TO +1.

CALL 'GTINFLNM' USING PARSED-FIELDS
                                        WS-FIELD-NAME
                                        WS-FIELD-LENGTH
       ON EXCEPTION
           DISPLAY 'ERROR ON GTINFLNM CALL'
END-CALL.

MOVE WS-FIELD-NAME                   TO WS-ARRAY.
MOVE WS-FIELD-LENGTH                 TO WS-DATA-LENGTH.
PERFORM B1000-MOVE-DATA-TO-OUTPUT.

MOVE WS-MAX-DATA-LENGTH              TO WS-DATA-LENGTH.

CALL 'GTFLDASC' USING PARSED-FIELDS
                      WS-FIELD-NAME
                      WS-INDEX
                      WS-ARRAY
                      WS-DATA-LENGTH
       ON EXCEPTION
           DISPLAY 'ERROR ON GTFLDASC CALL'
END-CALL.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                         MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                         TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                         ADD +1 TO WS-OUTPUT-LEN
                         SET OUTPUT-IDX UP BY +1
END-PERFORM.
```

```
MOVE SPACES                          TO WS-ARRAY.
MOVE SPACES                          TO WS-FIELD-NAME.
MOVE +0                              TO WS-FIELD-LENGTH.

CALL 'GTOTFLNM' USING PARSED-FIELDS
                      WS-FIELD-NAME
                      WS-FIELD-LENGTH
       ON EXCEPTION
           DISPLAY 'ERROR ON GTOTFLNM CALL'
END-CALL.

MOVE WS-FIELD-NAME                   TO WS-ARRAY.
MOVE WS-FIELD-LENGTH                 TO WS-DATA-LENGTH.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                          MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                          TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                          ADD +1 TO WS-OUTPUT-LEN
                          SET OUTPUT-IDX UP BY +1
END-PERFORM.

MOVE SPACES                          TO WS-ARRAY.
MOVE WS-MAX-DATA-LENGTH              TO WS-FILED-LENGTH.

CALL 'GTINFLST' USING PARSED-FIELDS
                      WS-FIELD-DATA
                      WS-FIELD-LENGTH.

MOVE WS-FIELD-DATA                   TO WS-ARRAY.
MOVE WS-FIELD-LENGTH                 TO WS-DATA-LENGTH.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                          MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                          TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                          ADD +1 TO WS-OUTPUT-LEN
                          SET OUTPUT-IDX UP BY +1
```

```
END-PERFORM.

MOVE SPACES                          TO WS-ARRAY.
MOVE +0                              TO WS-FIELD-LENGTH.
MOVE SPACES                           TO WS-FIELD-DATA.

CALL 'GTINFLDT' USING PARSED-FIELDS
                      WS-FIELD-DATA
                      WS-FIELD-LENGTH.

MOVE WS-FIELD-DATA                    TO WS-ARRAY.
MOVE WS-FIELD-LENGTH                  TO WS-DATA-LENGTH.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                          MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                          TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                          ADD +1 TO WS-OUTPUT-LEN
                          SET OUTPUT-IDX UP BY +1
END-PERFORM.

MOVE SPACES                          TO WS-ARRAY.

CALL 'GTINFLTP' USING PARSED-FIELDS
                      WS-FIELD-TYPE.

MOVE WS-FIELD-TYPE                    TO WS-FLD-TYPE.
MOVE WS-FLD-TYPE                      TO WS-ARRAY.
MOVE +9                               TO WS-DATA-LENGTH.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                          MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                          TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                          ADD +1 TO WS-OUTPUT-LEN
                          SET OUTPUT-IDX UP BY +1
END-PERFORM.
```

```
MOVE SPACES                          TO WS-ARRAY.
MOVE WS-MAX-DATA-LENGTH              TO WS-FILED-LENGTH.
MOVE SPACES                          TO WS-FIELD-DATA.

CALL 'GTINFLLN' USING PARSED-FIELDS
                      WS-FIELD-LENGTH.

MOVE WS-FIELD-LENGTH                 TO WS-FLD-TYPE.
MOVE WS-FLD-TYPE                     TO WS-ARRAY.
MOVE +9                              TO WS-DATA-LENGTH.

PERFORM
        VARYING WS-ARRAY-SUB FROM 1 BY 1
        UNTIL WS-ARRAY-SUB > WS-DATA-LENGTH
        OR WS-OUTPUT-LEN > WS-MAX-DATA-LENGTH
                          MOVE WS-ARRAY-BYTE (WS-ARRAY-SUB)
                          TO WS-OUTPUT-BYTE(OUTPUT-IDX)
                          ADD +1 TO WS-OUTPUT-LEN
                          SET OUTPUT-IDX UP BY +1
END-PERFORM.

MOVE SPACES                           TO WS-ARRAY.
MOVE WS-OUTPUT-LEN                    TO OUTPUT-LENGTH.
MOVE WS-OUTPUT-FIELD                  TO OUTPUT-FIELD.
MOVE +0                               TO RET-CODE.

DISPLAY ' OUTPUT FIELD ON EXIT IS: '
        WS-OUTPUT-FIELD (1:OUTPUT-LENGTH).
DISPLAY 'OUTPUT LENGTH ON EXIT IS: ' OUTPUT-LENGTH.
End Program "TESTCOB1".
```

Sample Program for COBOL User Exits Using DB2

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. "TESTCOB3".
```

```
 AUTHOR. NNSY - NEW ERA OF NETWORKS.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*SOURCE-COMPUTER. IBM-370 WITH DEBUGGING MODE.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  BYVAL              PIC 9(4)      VALUE 99 USAGE IS BINARY.
 01  HOLD-VALUE         PIC X(20)     VALUE SPACES.
 01  FIELD-NAME         PIC X(10)     VALUE Z'JimField2'.
 01  FIELD-INDEX        PIC S9(9)     VALUE +0 COMP.
 01  WS-OUTFIELD        PIC X(20).
 01  WS-OUTFIELD-LENGTH PIC S9(9)     VALUE +20 COMP.
 01  WS-INFIELD         PIC X(20).
 01  WS-INFIELD-LENGTH  PIC S9(9)     VALUE +20 COMP.
     EXEC SQL
         INCLUDE SQLCA
     END-EXEC.
     EXEC SQL
         INCLUDE NNFLITRL
     END-EXEC
 01 SQLCODE-VALUES.
    05 SQL-SUCCESSFUL   PIC S9(9)     COMP VALUE 0.
    05 SQL-RECORD-NOT-FOUND
                        PIC S9(9)     COMP VALUE
+100.
    05 SQLCODE-DISPLAY  PIC 9(9).
 01 WORK-OUTFIELD.
    05 FIRST-PART       PIC X(8).
    05 FILLER           PIC X(2)      VALUE '::'.
    05 SECOND-PART      PIC X(3).
    05 FILLER           PIC X(2)      VALUE '::'.
    05 THIRD-PART       PIC X(4).
   05 END-PART          PIC S9(4)     VALUE +0 COMP.
LINKAGE SECTION.
01  SESSION-POINTER       USAGE POINTER.
01  PARSED-FIELDS-POINTER USAGE POINTER.
01  OUTPUT-FIELD-LENGTH   PIC S9(9)   BINARY.
01  OUTPUT-FIELD          PIC X(75).
01  RETURN-CODE-VALUE     PIC S9(9)   BINARY.
PROCEDURE DIVISION USING BY REFERENCE SESSION-POINTER
                        PARSED-FIELDS-POINTER
                        OUTPUT-FIELD-LENGTH
```

```
                          OUTPUT-FIELD
                          RETURN-CODE-VALUE.
    DISPLAY 'WE ARE IN TESTCOB3'.
    DISPLAY 'PARSED-FIELDS-POINTER = ' PARSED-FIELDS
POINTER.
    MOVE OUTPUT-FIELD TO WS-OUTFIELD.
    DISPLAY 'WS-OUTFIELD = ' WS-OUTFIELD.
    CALL 'GTINFLDT' USING
                          PARSED-FIELDS-POINTER
                          WS-INFIELD
                          WS-INFIELD-LENGTH.
    DISPLAY 'BACK FROM GTINFLDDT : DATA = ' WS-INFIELD
            ' WS-INFIELD-LENGTH = ' WS-INFIELD-LENGTH.
    MOVE WS-INFIELD TO FIRST-PART.
    CALL 'GTFLDASC' USING
                          PARSED-FIELDS-POINTER
                          FIELD-NAME
                          FIELD-INDEX
                          WS-OUTFIELD
                          WS-OUTFIELD-LENGTH.
    DISPLAY 'BACK FROM GTFLDASC : DATA = ' WS-OUTFIELD
            ' WS-OUTFIELD-LENGTH = ' WS-OUTFIELD-LENGTH.
    MOVE WS-OUTFIELD TO SECOND-PART.
    EXEC SQL
        DECLARE C2 CURSOR  FOR
        SELECT *   FROM NN52QA2.NNF_LITRL
    END-EXEC
    EXEC SQL
      OPEN C2
    END-EXEC.
    IF SQLCODE = SQL-SUCCESSFUL
        EXEC SQL
          FETCH C2 INTO :DCLNNF-LITRL
        END-EXEC
    ELSE
        DISPLAY 'OPEN CURSOR FAILED : SQLCODE = '
SQLCODE
        DISPLAY 'SQLERRMC = ' SQLERRMC.
    MOVE LITRL-NAME-TEXT TO HOLD-VALUE.
    DISPLAY 'BACK FROM DB2 : VALUE = ' HOLD-VALUE.
    MOVE LITRL-NAME-TEXT TO THIRD-PART.
    EXEC SQL
```

```
            CLOSE C2
           END-EXEC.
           COMPUTE OUTPUT-FIELD-LENGTH = WS-INFIELD-LENGTH
                    + WS-OUTFIELD-LENGTH + LITRL-NAME-LEN + 4.
           MOVE WORK-OUTFIELD TO OUTPUT-FIELD.
           DISPLAY 'STRUNG OUTFIELD = ' OUTPUT-FIELD.
            MOVE +0 TO RETURN-CODE-VALUE.
        END PROGRAM "TESTCOB3".
```

Sample Program for C++ User Exits Using DB2

```
#include <stdlib.h>
#include <string.h>
#include <INFR/Streams.h>
// Include files for database access
#include <OLD/ses.h>
#include <OT/NNOT.h>
#include <SES/NNSesDBBase.h>
#include <SES/NNSesDB2.h>
#include <TOOLS/NNAlert.h>
// Include files for formatter
#include <FMTR/nnexit.h>
#include <FMTR/formatter.h>
#include <time.h>
NNExitRet
ReadDB(const DbmsSession &rSession, const NNParsedFields &rFields) {
    NNExitRet oER;
    SQLRETURN results ;
    // In order to use our DB2 connection, we must get the HDBC
handle.
    // This involves getting to out NNSesDB2 session
    DbmsSession* db = (DbmsSession*) &rSession ;
    NNSesDB2 *OurSession = (NNSesDB2*) db->getSession()
    HDBC  myHdbc = OurSession->getHDBC() ;
    cout << "Back from getHDBC" << endl ;
    HENV myHenv = OurSession->getHENV() ;
```

```
    SQLCHAR SqlState [5] ;
   SQLINTEGER NativeError ;
   SQLCHAR    ErrorMessage [100] ;
   SQLSMALLINT ErrorMessageSz = 100 ;
   SQLSMALLINT Avail ;
   // Save the SQLID on entry
   char EntrySQLID [32] ;
   strcpy (EntrySQLID, OurSession->getUserID()) ;
   // Now,  get a statement handle
   HSTMT myHstmt ;
   cout << "About to go to SQLAllocStmt" << endl ;
   results = SQLAllocStmt (myHdbc, &myHstmt) ;
   cout << "Back from SQLAllocStmt" << endl ;
   if (results == SQL_ERROR)
   {
     oER.SetError(NN_ERSTATUS_ERROR, "Attempt to allocate a statement
handle failed") ;
     return  oER ;
   }
   // OK, Now we have our handle, set the SQLID
   char sql [100] = " SET CURRENT SQLID = 'NN52DEV1'";
   cout << "SQL built to set the SQLID = " << sql << endl ;
   results = SQLExecDirect (myHstmt, (unsigned char*) sql,
strlen(sql))
   if (results == SQL_ERROR)
   {
     oER.SetError (NN_ERSTATUS_ERROR, "ReadDB failed to set the
SQLID")
     SQLFreeStmt (myHstmt, SQL_DROP) ;
     return oER ;
   }
   // Now issue our DataBase call
   SQLCHAR *sqlSelect = (SQLCHAR*) "SELECT COUNT(*) FROM NNF_FMT" ;
   results = SQLExecDirect (myHstmt, sqlSelect, SQL_NTS) ;
   if (results == SQL_ERROR)
   {
     oER.SetError (NN_ERSTATUS_ERROR, "ReadDB failed to get the
count")
     SQLFreeStmt (myHstmt, SQL_DROP) ;
     return oER ;
   }
   // Now fetch the result
```

```
   results = SQLFetch (myHstmt) ;
   if (results == SQL_ERROR)
   {
     oER.SetError (NN_ERSTATUS_ERROR, "ReadDB failed on the FETCH") ;
     SQLFreeStmt (myHstmt, SQL_DROP) ;
     return oER ;
   }
   // Now format the results
   long Count ;
   SQLINTEGER  dataLen ;
   results = SQLGetData (myHstmt, 1, SQL_C_LONG, &Count, sizeof(Count)
   cout << "The result of our SQL call was : Count = " << Count << end
   oER = long(Count);
   // Now reset the SQLID
   SQLFreeStmt (myHstmt, SQL_DROP) ;
   cout << "OurSession DBName is : " << OurSession->getDBname() <<
endl
   strcpy ( sql, " SET CURRENT SQLID = '") ;
   strcat ( sql, OurSession->getDBname() ) ;
   strcat ( sql, "' ") ;
   cout << "SQL build to reset the SQLID = " << sql << endl ;
   results = SQLExecDirect (myHstmt, (unsigned char*) sql,
strlen(sql));
   if (results == SQL_ERROR)
   {
     SQLError (myHenv, myHdbc, myHstmt,SqlState, &NativeError,
ErrorMe
     cout << "Our error when trying to reset the SQLID : SqlState = "
<< SqlState
          << "ErrorMessage = " << (char *) ErrorMessage
          << " Native Error = " << NativeError << endl ;
     oER.SetError (NN_ERSTATUS_ERROR, "ReadDB failed to reset the
SQLID") ;
     SQLFreeStmt (myHstmt, SQL_DROP) ;
     return oER ;
   }
   // Now release the stmt handle and return
   SQLFreeStmt (myHstmt, SQL_DROP) ;
    return    oER;
}
extern "C" void
NNGetUserExitFuncPtrs(
```

```
 char*,,,acFuncName,
 NN_EXIT_FUNC_t,,&rUEptr,
 NN_EXIT_CLEANUP_FUNC_t,&rUEClUpPtr) {
    if(strcmp(acFuncName, "ReadDB") == 0) {
       rUEptr = ReadDB;
       rUEClUpPtr = NULL;
    }
    else {
      rUEptr = NULL;
      rUEClUpPtr = NULL;
    }
}
```

Appendix B
Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this document to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

```
IBM United Kingdom Laboratories,
Mail Point 151,
Hursley Park,
Winchester,
Hampshire,
England,
SO21 2JN.
```

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms.

You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks and Service Marks

The following, which appear in this book or other Rules and Formatter Support for WebSphere MQ Integrator books, are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

```
WebSphere
z/OS
MQSeries
AIX
DB2
IBM
```

New Era of Networks Rules and New Era of Networks Formatter are trademarks of New Era of Networks, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

UNIX is a registered trademark in the United States and/or other countries licensed exclusively through The Open Group.

Other company, product, or service names may be the trademarks or service marks of others.

Index

U

Sending your comments to IBM
Rules and Formatter Support for WebSphere MQ Integrator
COBOL User Exits Supplement

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book only and the way in which the information is presented.

To request additional publications or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form
- By fax:
  - From outside the U.K., use your international access code followed by 44 1962 870229
  - From within the U.K., use 01962 870229

Electronically, use the appropriate network ID:

- IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
- IBMLink: HURSLEY(IDRCF)
- Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic number to which your comment applies
- Your name / address / telephone number / fax number / network ID

Readers' Comments
Rules and Formatter Support for WebSphere MQ Integrator
COBOL User Exits Supplement

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name Address

Company or organization

Telephone Email