MQSeries® Integrator for OS/390®

Program Supplement

Version 1.1.1

GC34-5875-00

Note: Before using this information, and the product it supports, be sure to read the general information under *Notices* on page 125**.**

First edition (December 1999)
This edition applies to IBM® MQSeries Integrator, Version 1.1 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

The MQSeries Integrator CSD provides several new product features. This supplement provides information on how to use these new features. It is organized into the following chapters:

- Chapter 1, *Introduction*, provides an outline of the contents of this supplement, the documentation set, and technical support information.
- Chapter 2, *Product Enhancements*, provides two new Formatter APIs, one Formatter Management API, and a supported NCF.
- Chapter 3, *Developing COBOL User Exits*, describes how COBOL user exits can customize NEONFormatter, provides procedures for creating COBOL user exits, and lists the COBOL wrapped versions of NEONFormatter APIs by class.
- Chapter 4, *Testing Formats*, describes how to test NEONFormatter using a COBOL driver program. It provides a summary of the API calls, a sample source code flow chart of the main calls, and detailed API description.
- Chapter 5, *Testing Rules*, describes how to test NEONRules using a COBOL driver program. It provides a summary of the API calls, a sample source code flow chart of the main calls, and detailed API descriptions.
- Chapter 6, *Compiling the Sample COBOL Programs*, illustrates how a sample COBOL application calls MQSeries Integrator APIs, provides procedures and sample JCL to compile and link the sample COBOL programs, and procedures for running and troubleshooting the COBOL test programs.
- Appendix A, *COBOL Sample Programs*, provides complete program samples of MSGTSTCB, RULTSTCB, and TESTCOB1.

- Appendix B, *Notices*, provides IBM trademark and service mark information.

Documentation Set

The MQSeries Integrator for OS/390 documentation includes:

- ***Installation and Configuration Guide***
- ***User's Guide***
- ***System Management Guide***
- ***Programming References***
  - ***Application Development Guide***
  - ***Programming Reference for NEONFormatter***
  - ***Programming Reference for NEONRules***
- ***Program Supplement***

Year 2000 Readiness Disclosure

MQSeries Integrator, when used in accordance with its associated documentation, is capable of correctly processing, providing, and/or receiving date information within and between the twentieth and twenty-first centuries, provided that all products (for example, hardware, software, and firmware) used with this IBM program properly exchange accurate date information with it.

Customers should contact third party owners or vendors regarding the readiness status of their products.

IBM reserves the right to update the information shown here. For the latest information regarding levels of supported software, refer to:
http://www.software.ibm.com/ts/mqseries/platforms/supported.html

For the latest IBM statement regarding Year 2000 readiness, refer to:
http://www.ibm.com/ibm/year2000/

Before You Contact Technical Support

If you have difficulty executing one of the MQSeries Integrator programs, analyze your environment using the following steps. Be prepared to send the listed information and files to technical support.

1. Has this program ever worked in your environment?

   If so, identify what has changed.

2. Check the values specified in the SQLSVSES (DD-name SQLSVSES) file that the failing job is using to make sure it refers to an existing DB2 subsystem and an existing DB2 database within that subsystem.

3. Check the values specified in the CLIINI (DD-name DSNAOINI) file that the failing job is using to make sure it refers to an existing DB2 subsystem and an existing DB2 database within that subsystem.

4. Check whether the System Affinity is causing your job to execute on a system that does not contain the DB2 subsystem, MQSeries queue manager, or IBM datasets that MQSeries Integrator is trying to access.

5. In the CLIINI file (DD-name DSNAOINI), edit the following line:

   ```
   CLITRACE=0
   ```

   Change it to:

   ```
   CLITRACE=1
   ```

   Rerun your job. The CLITRACE produced (DD-name CLITRACE) is invaluable in diagnosing problems between the DB2 database and the MQSeries Integrator application. Your JCL should have a DD-statement that defines CLITRACE to either a disk file or SYSOUT class. This file is required by technical support to diagnose problems.

Note:
It is assumed that the MQSeries Integrator Plan has been bound, and you are granted execute authority on it.

6. Examine all files produced by MQSeries Integrator for error or informational messages. Some error messages are written to SYSOUT, some to SYSPRINT, and some to STATLOG.
7. Look for Operating System messages that may indicate why the job has failed, such as missing files, no room to log messages (E-37, B-37 type failures), full queue conditions, and so on.
8. If failing to put or get from an MQSeries queue, make sure the queue is enabled for sharing:

   ```
   Permit shared access . . . . Y  Y=Yes,N=No
   Default share option . . . . S E=Exclusive,S=Shared
   ```

9. If the problem is related to poor Rules daemon performance, check the values of the timers specified in the input stream (DD-name SYSIN) file of the RULENG job. Setting these timers too high can result in poor performance of the Rules Engine.

When contacting technical support be prepared to send the following information via email or ftp:

- The complete listing of your jobs execution, including SYSOUTs, SYSPRINTs, STATLOG, JESMSGS, and so on.
- The contents of the CLITRACE file
- Any dump files produced (CEEDUMP or SYSUDUMP)
- Your site's SQLSVSES file
- Your site's CLIINI file

Chapter 2

Product Enhancements

This chapter describes new product features included in the CSD for MQSeries Integrator. It includes two new NEONFormatter APIs and one NEONFormatter Management API:

- GetCurrOutFldLength
- GetCurrOutFldData
- NNFmtRmv()

Formatter APIs

The following section details the C++ version of new NEONFormatter APIs. For more information on NEONFormatter APIs, see ***Programming Reference for NEONFormatter***, p. 81.

NNParseField

These C++ Formatter APIs are for the NNParseField class. For the correlating COBOL wrapped version, see *GTOTFLDT* on page 29.

GetCurrOutFldData

Returns the current value of the input field data as it exists prior to invoking a user exit. Any transformation resulting from output operations that occurs prior to the user exit is reflected.

Syntax

```
const char * GetCurrOutFldData() const
```

Parameters

none

Return Value

Returns the current value of the input field data.

GetCurrOutFldLength

Returns the current length of the output field data as it exists prior to invoking a user exit. Any transformation resulting from output operations that occurs prior to the user exit is reflected.

Syntax

```
const int GetCurrOutFldLength() const
```

Parameters

none

Return Value

Returns the current length of the input field data.

Formatter Management APIs

The following section details the C++ version of a new NEONFormatter Management API. For more information on NEONFormatter APIs, see ***Programming Reference for NEONFormatter***, p. 81.

NNFmtRemover

This C++ Formatter API is for the NNFmtRemover class. This API has no corresponding COBOL wrapped version.

NNFmtRmv

NNFmtRmv removes input or output formats and all associated components that are not being used by another Formatter component from the database.

Syntax

```
<remove>[ -fN format_name]+
```

Parameters

Name	Type	Input/ Output	Description
format_ name	n/a	Input	Sends the name of the format to be removed.

Remarks

Maintains referential integrity by verifying that only the components that are not being used by other Formatter components are removed. The associate components include: fields, parse controls, literals, and output controls.

Return Value

Returns zero (0) if successful; 1 if failed.

Examples

```
NNFmtRmv remove -fN flat_format_1
NNFmtRmv remove -fN flat_format -fN flat_format_2
NNFmtRmv remove -fN compound_format
```

Chapter 3

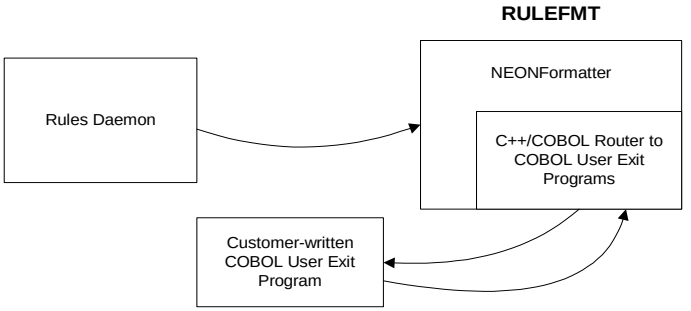
Developing COBOL User Exits

This chapter describes how COBOL user exits can be used to customize NEONFormatter, provides procedures for creating COBOL user exits, and lists the COBOL wrapped versions of NEONFormatter APIs by class. For general information on user exits, see the ***System Management Guide***.

Customizing NEONFormatter

By creating a NEONFormatter user exit, you can externally customize NEONFormatter 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 NEONFormatter, such as adding field values within a repeating structure, or
- call MQSeries Integrator functions by statically linking the user exit APIs to the user application.



This diagram illustrates the process. The Rules daemon gets a message from the input queue and sends it to NEONFormatter to parse the message. NEONFormatter reformats the parsed message contents and places it onto an output message format. NEONFormatter calls and loads customer-written COBOL user exits to customize the contents of the output message. The output message and program control are then returned to NEONFormatter.

Combining COBOL with C++

When combining COBOL and C++ programs to create user exits:

- COBOL statically calls C functions,
- Only one C/C++ load module can be linked with a COBOL program,
- C/C++ dynamically calls C++ functions.

Creating COBOL User Exits

COBOL user exits exist as standalone COBOL programs and must have unique names of eight bytes or less. The name of a user exit must be equal to the name of the COBOL program in which it is implemented. Because the name is not case-sensitive, the names CobolUE and COBOLUE both identify the same user exit, implemented in COBOL program COBOLUE.

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

Note:
For more information on creating user exits in C++, see ***Programming Reference for NEONFormatter***.

To create COBOL user exits:

1. Write the user exit program in COBOL, and define the LINKAGE SECTION of the user exit program as follows.

   LINKAGE SECTION

   All parameters from NEONFormatter are passed by reference to the user application through the LINKAGE SECTION.

```
LINKAGE SECTION.
    01  SESSION-POINTER        POINTER.
    01  PARSED-FIELDS          POINTER.
    01  OUTPUT-LENGTH          PIC  S9(09)COMP.
    01  OUTPUT-FIELD           PIC  X(2000).
    01  RET-CODE               PIC  S9(09)COMP.
```

Parameter	Input/ Output	Description
SESSION-POINTER	Input	Sends a pointer to the DBMS session.
PARSED-FIELDS	Input	Sends the data to be transformed to the user exit

Parameter	Input/ Output	Description
OUTPUT-LENGTH	Output	Returns the length of the output field passed back to NEONFormatter.
OUTPUT-FIELD	Output	Returns the data that was transformed by the user exit code to NEONFormatter.
RET-CODE	Output	Returns one of the following values to NEONFormatter. 0 = successful 16 = failed

2. Compile and link the user exit code as illustrated in the following sample JCL:

```
//* <insert a valid JOBCARD for your site>
//*************************************************************
//*                                                           *
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.        *
//* All Rights Reserved.                                      *
//*                                                           *
//* Release 4.1.1                                             *
//*************************************************************
//PROCLIST JCLLIB ORDER=(<smphlq>.SNEOPROC,SYS1.PROCLIB,SYS2.PROCLIB)
//*
//COMPILE  EXEC COBCOMP,LIBPRFX=CEE,MEMBER=TESTCOB1,
//       OUTFILE='<smphlq>.SNEOLOAD'
//COMPILE.SYSIN DD DSN=<smphlq>.SNEOSRCE(TESTCOB1),DISP=SHR
//LKED.SYSLMOD DD UNIT=SYSALLDA
//LKED.SYSIN   DD *
  NAME  TESTCOB1(R)
```

3. If you use both C++ and COBOL user exits, define the following as the default in the NNUESTUB member in the <hlq>.SNEOCPP library:

```
*pUEptr = 0

*pEUEIUpPtr = 0
```

4. Test the COBOL user exit program using MSGTSTCB. For more information, see *Testing Formats* on page 31.

API Reference for COBOL Wrappers

This section details the COBOL wrapped user exit APIs. COBOL user exit APIs allow the user application to access NEONRules and NEONFormatter from a COBOL application. The wrappers are listed by class and can be found in the SNEOLKED library.

NNParsedField

The following is a list of the COBOL wrapped APIs for the NNParsedField class. For more information on NNParsedFields class member functions, see ***Programming Reference for NEONFormatter, p. 139.***

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.

Working Storage

```
01  PARSED-FIELDS                  POINTER.
01  INPUT-FIELD-NAME               PIC X(33).
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.
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.

Parameter	Input/ Output	Description
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(33).
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 NEONFormatter as the output field value.

In order to use the sample exit, you must create a user exit named TESTCOB1 using the NEONFormatter 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, the Formatter will call program TESTCOB1 which will then execute using the parsed fields from your message as input.

The source and load for TESTCOB1 can be found in the <hlq>.SNEOSRCE and <hlq>.SNEOLOAD libraries. To compile TESTCOB1, use member TESTCOBC in the <hlq>.SNEOJCL library.

For the complete program sample, see *TESTCOB1 Sample Program* on page 119.

Chapter 4

Testing Formats

This chapter describes how to test NEONFormatter using a COBOL driver program. It provides a summary of the API calls, a sample source code flow chart of the main calls, and detailed API descriptions.

MSGTSTCB

MSGTSTCB is the COBOL test program. Its components are:

- a COBOL driver program,
- API wrappers called by the COBOL program that interface between the COBOL program and the C++ modules,
- C++ modules that exercise NEONFormatter APIs.

Notes:

For the complete sample program, see *MSGTSTCB Sample Program* on page 95.

For sample JCL to run program MSGTSTCB, see *Running the COBOL Test Programs* on page 89.

For more information on NEONFormatter, see the ***System Management Guide***.

MSGTSTCB API Calls

The following list summarizes the flow of API calls in the order in which they are used in MSGTSTCB.

STRTSESS creates the DBMS session.

```
CALL 'STRTSESS' USING SESSION-PTR
                      SESSION-NAME
                          DB2-TYPE.
```

STRTFMTR instantiates NEONFormatter.

```
CALL 'STRTFMTR' USING SESSION-PTR
                       FORMATR-PTR.
```

ADNPTMSG adds the Message.

```
CALL 'ADNPTMSG' USING FORMATR-PTR
                     IN-FORMAT-PTR
                      INPUT-RECORD
                        IN-REC-LEN.
```

ADOPTFMT assigns an output format.

```
CALL 'ADOPTFMT' USING FORMATR-PTR
                    OUT-FORMAT-PTR.
```

PARSEMSG deconstructs the message into its component parts.

```
CALL 'PARSEMSG' USING FORMATR-PTR
                       RETURN-CD.
```

REFORMAT translates the input message into an output message.

```
CALL 'REFORMAT' USING FORMATR-PTR
                       RETURN-CD.
```

Specialized data retrieval can be done at this time. The program checks to see if the formats are compound or flat, and data is retrieved accordingly.

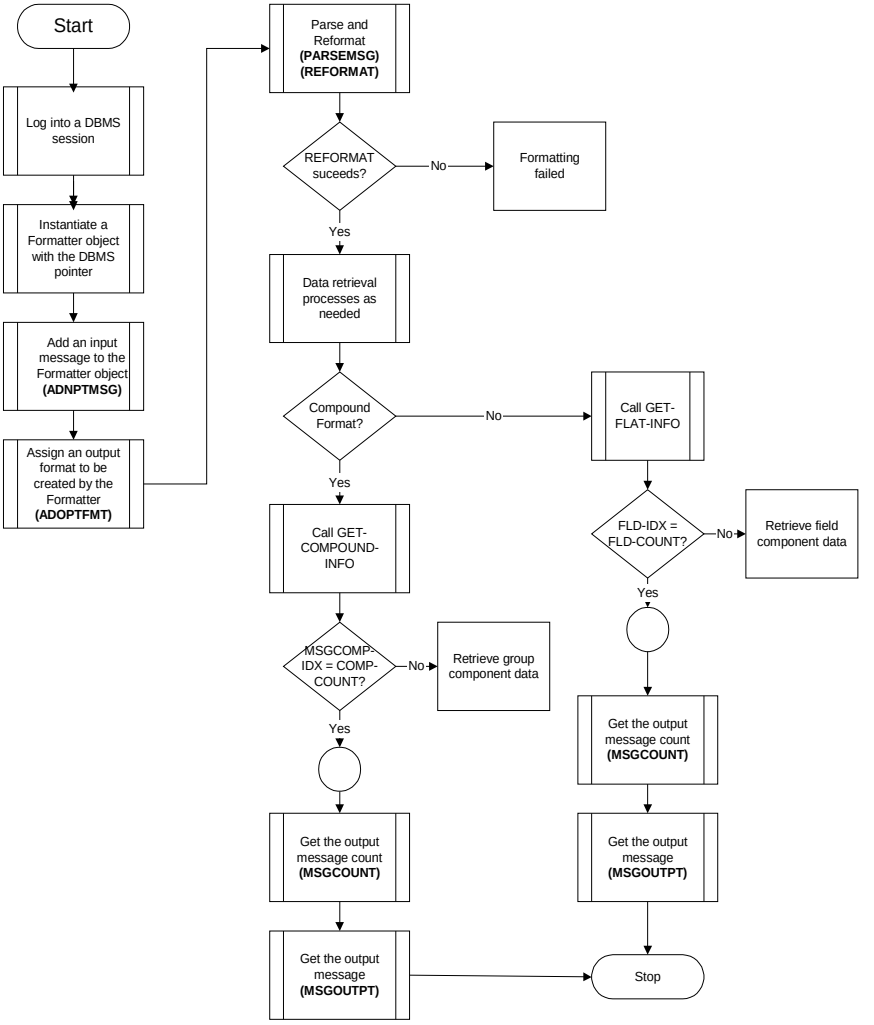
MSGCOUNT returns the output message count.

```
CALL 'MSGCOUNT' USING MSG-CNT
              OUT-MESSAGE-PTR
             OUT-MESSAGE-HOLD.
```

MSGOUPT retrieves the output message and writes it to a file.

```
CALL 'MSGOUTPT' USING OUT-MESSAGE-HOLD
                      MESSAGE-VALUE-OUT
                       MESSAGE-SIZE-OUT.
WRITE OUTPUT-RECORD FROM MESSAGE-VALUE-OUT.
```

MSGTSTCB Flow of Calls



API Reference for COBOL Wrappers

This section details the COBOL wrapped version of the NEONFormatter APIs that are contained in the sample MSGTSTCB program. The wrappers are listed by class.

For more information on NEONFormatter APIs, see ***Programming Reference for NEONFormatter***, p. 81.

DbmsSession

The following is a COBOL wrapped version of the DbmsSession class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

STRTSESS

OpenDbmsSession() creates the Dbms session.

Working Storage Section

```
01  SESSION-PTR            POINTER   VALUE NULL.
01  SESSION-NAME           PIC X(08) VALUE 'SESSION-NAME'.
01  DATABASE-VALUES.
     05  DB2-TYPE          PIC S9(9) VALUE 5 USAGE IS BINARY.
```

COBOL Calling Portion

```
CALL 'STRTSESS' USING SESSION-PTR
                      SESSION-NAME
                          DB2-TYPE.
```

Usage Pointers	Input/ Output	Description
SESSION-PTR	Output	Returns the address of the session.

Usage Pointers	Input/ Output	Description
SESSION-NAME	Input	Sends the name of DBMS session-defined in SQLSVSES file.
DB2-TYPE	Input	Sends the type of database used.

Formatter

The following is a list of the COBOL wrapped version of the Formatter class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

STRTFMTR

Formatter instantiates NEONFormatter by sending the current session pointer and a NULL NEONFormatter pointer.

Working Storage Section

```
01  SESSION-PTR            POINTER.
01  FORMATR-PTR            POINTER VALUE NULL.
```

COBOL calling portion

```
CALL 'STRTFMTR' USING SESSION-PTR
                      FORMATR-PTR
```

Usage Pointers	Input/ Output	Description
SESSION-PTR	Input	Sends the address of the current session.
FORMATR-PTR	Output	Returns the address of the NEONFormatter instance.

ADNPTMSG

AddInputMessage() stores a copy of the input message and input format name within the current NEONFormatter instance. The format name is validated after parse() is called through PARSEMSG or REFORMAT.

Working Storage Section

```
01  FORMATR-PTR          POINTER.
01  IN-FORMAT-PTR        PIC X(33) VALUE 'INPUT-FORMAT-NAME'.
01  INPUT-RECORD.
    05  INPUT-REC        PIC X(01)
                         OCCURS 1 TO 19624 (adjust for input
                                            file)
                         DEPENDING ON IN-REC-LEN
                         INDEXED BY IN-IDX.
```

COBOL calling portion

```
CALL 'ADNPTMSG' USING FORMATR-PTR
                      IN-FORMAT-PTR
                       INPUT-RECORD
                         IN-REC-LEN.
```

Usage Pointers	Input/ Output	Description
FORMAT-PTR	Input	Sends the address of the current NEONFormatter instance.
IN-FORMAT-PTR	Input	Sends the address of the input format name.
INPUT-RECORD	Input	Sends the address of the message to be formatted.
IN-REC-LEN	Input	Sends the length of the message buffer.

ADOPTFMT

AddOutputFormat() retrieves the specified output format from the database to create the output message.

Working Storage Section

```
01  FORMATR-PTR             POINTER.
01  OUT-FORMAT-PTR          PIC X(33) VALUE SPACES.
```

COBOL calling portion

```
CALL 'ADOPTFMT' USING FORMATR-PTR
                      OUT-FORMAT-PTR.
```

Usage Pointers	Input/ Output	Description
FORMAT-PTR	Input	Sends the address of the current NEONFormatter instance.
OUT-FORMAT-PTR	Input	Sends the address of the output format name.

PRELDMSG

PreloadInFormat() immediately preloads an input format into memory. If you do not use this function call, input formats automatically get loaded from the database during a call to PARSEMSG or REFORMAT.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
01  IN-FORMAT-PTR          PIC X(33) VALUE 'INPUT-FORMAT-NAME'.
```

COBOL calling portion

```
CALL 'PRELDMSG' USING FORMATR-PTR
                      IN-FORMAT-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
IN-FORMATR-PTR	Input	Sends the address of the input format name.

PRELDFMT

PreloadOutFormat() immediately pre-loads the output format into memory. If you do not use this function call, output formats automatically get loaded from the database during a call to REFORMAT.

Working Storage Section

```
01  FORMATR-PTR             POINTER.
01  OUT-FORMAT-PTR          PIC X(33)VALUE 'OUTPUT-FORMAT-NAME'.
```

COBOL calling portion

```
CALL 'PRELDFMT' USING FORMATR-PTR
                      OUT-FORMAT-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
OUT-FORMATR-PTR	Input	Sends the address of the output format name.

PARSEMSG

parse() deconstructs the message into its component fields. This enables user applications to process individual field data by calling message access APIs, such as GETASCII and GTFLDVAL.

Working Storage Section

```
01  FORMATR-PTR           POINTER.
01  RETURN-CD             PIC S9(9) VALUE 0 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'PARSEMSG' USING FORMATR-PTR
                      RETURN-CD.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
RETURN-CD	Output	Returns a value indicating success or failure.

Return Values

If the parse is successful, the function returns (1); if it fails, it returns zero (0).

REFORMAT

reformat() translates input messages into output messages by automatically calling the parse() API.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
01  RETURN-CD              PIC S9(9) VALUE 0 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'REFORMAT' USING FORMATR-PTR
                      RETURN-CD.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
RETURN-CD	Output	Returns a value indicating success or failure.

Return Values

If the reformat is successful, the function returns (1); if it fails, it returns zero (0).

GETASCII

Based on the field name, GetFieldAscii() enables access to the field contents, returning a NULL-terminated representation of the field content.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
01  FIELD-NAMES.(depending on number of fields in message)
    05  FIELD-ONE          PIC X(33)  VALUE 'FIELD ONE NAME'.
    05  FIELD-TWO          PIC X(33)  VALUE 'FIELD TWO NAME'.
    05  FIELD-THREE        PIC X(33)  VALUE 'FIELD THREE NAME'.
    05  FIELD-FOUR         PIC X(33)  VALUE 'FIELD FOUR NAME'.
01  SEQ-NUM                PIC 9(9)   VALUE 00 USAGE IS BINARY.
01  GET-ASCII-MESSAGE.
    05  ASCII-MESSAGE      PIC X(806) VALUE SPACES.
01  STRING-LENGTH          PIC 9(9)   VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'GETASCII' USING FORMATR-PTR
                      FIELD-ONE
                        SEQ-NUM
                   ASCII-MESSAGE
                   STRING-LENGTH.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
FIELD-ONE	Input	Sends the name of the field.
SEQ-NUM	Input	Sends an index specifying which field to reference, if a field appears more than once in a message. This index starts at and defaults to zero (0), incrementing by one as the user moves from left to right through the input message buffer.

Usage Pointers	Input/ Output	Description
ASCII-MESSAGE	Output	Returns the field contents.
STRING-LENGTH	Output	Returns the length of the field.

Return Value

Returns a NULL-terminated representation of the field content.

GTASCITG

GetFieldAsciiByTag() returns the tag contents and tag length in bytes, using the tag name and tag index for fields appearing more than once on a message. This function must be called after PARSEMSG.

Working Storage Section

```
01  FORMATR-PTR             POINTER.
01  TAG-TWO                 PIC X(33) VALUE 'TAG NAME'.
01  TAG-LEN                 PIC 9(5)  VALUE ZERO.
01  TAG-NUM                 PIC 9(9)  VALUE 00 USAGE IS BINARY.
01  TAG-LENGTH              PIC 9(9)  VALUE 00 USAGE IS BINARY.
01  GET-ASCII-TAG-MESSAGE.
    05  ASCII-TAG-MESSAGE   PIC X(19624) VALUE SPACES.
01  GET-ASCII-TAG-TABLE.
    05  ASCII-TAG-TABLE     PIC X(01)
              OCCURS 1 TO 19624 TIMES
                 DEPENDING ON TAG-LEN.
```

COBOL calling portion

```
CALL 'GTASCITG' USING FORMATR-PTR
                      TAG-TWO
                      TAG-NUM
                ASCII-TAG-MESSAGE
                      TAG-LENGTH.
MOVE TAG-LENGTH TO TAG-LEN.
MOVE ASCII-TAG-MESSAGE TO GET-ASCII-TAG-TABLE.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
TAG-TWO	Input	Sends the name of the tag.

Usage Pointers	Input/ Output	Description
TAG-NUM	Input	Sends an index specifying which field to reference if a field appears more than once in a message. The index starts at zero (default), incrementing by one as you move from left to right through the input message buffer.
ASCII-TAG-MESSAGE	Output	Returns the value of the tag.
TAG-LEN	Output	Returns the length of the tagged message.

OUTMSCNT

GetOutMsgCount() returns the number of output message groups in the NEONFormatter object.

Working Storage Section

```
01  FORMATR-PTR              POINTER.
01  OUT-MSG-COUNT            PIC S9(9) VALUE 0 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'OUTMSCNT' USING FORMATR-PTR
                      OUT-MSG-COUNT.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
OUT-MSG-COUNT	Output	Returns the number of output message groups in the NEONFormatter object.

GETOTMSG

GetOutMsgGroup() returns a pointer to the output message group identified by the format name.

Working Storage Section

```
01  FORMATR-PTR          POINTER.
01  OUT-FORMAT-PTR       PIC X(33) VALUE 'Output-Format-Name'.
01  OUT-MESSAGE-PTR      POINTER VALUE NULL.
```

COBOL calling portion

```
CALL 'GETOTMSG' USING FORMATR-PTR
                      OUT-FORMAT-PTR
                     OUT-MESSAGE-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
OUT-FORMAT-PTR	Input	Sends the output format name.
OUT-MESSAGE-PTR	Output	Returns the address of the output message group. Must be set to NULL before a call.

PRSMSCNT

GetparsedInMsgCount() returns the number of input messages parsed by NEONFormatter using the NEONFormatter pointer. The number of messages returned should equal the number of input messages added by ADNPTMSG.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
01  PARSED-IN-NUM          PIC S9(9) VALUE 0 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'PRSMSCNT' USING FORMATR-PTR
                      PARSED-IN-NUM.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
PARSED-IN-NUM	Output	Returns the number of parsed input messages.

GTPRSMSG

GetParsedInMsg() returns a pointer to a parsed input message at the specified index. Starting at zero (0) for the first message, the index follows the order in which messages were added using ADNPTMSG.

Working Storage Section

```
01  FORMATR-PTR              POINTER.
01  PARSED-IN-NUM            PIC S9(9) VALUE 0 USAGE IS BINARY.
01  PARSEDMSG-PTR            POINTER VALUE NULL.
```

COBOL calling portion

```
CALL 'GTPRSMSG' USING FORMATR-PTR
                      MSG-IDX
                  PARSEMSG-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
MSG-IDX	Input	Sends the index of the parsed input message to return.
PARSEDMSG-PTR	Output	Returns the parsed input message at the specified index.

SETVALON

SetUserTypeValidationOn() turns on the user-defined type input field validation. This function sets the validation state of all fields defined by user-defined type; however, the validation state of individual fields cannot be set.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
```

COBOL calling portion

```
CALL 'SETVALON' USING FORMATR-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.

Default State

On

SETVALOF

SetUserTypeValidationOff() turns off the user-defined type input field validation. This function sets the validation state of all fields defined by user-defined type; however, the validation state of individual fields cannot be set.

Working Storage Section

```
01  FORMATR-PTR            POINTER.
```

COBOL calling portion

```
CALL 'SETVALOF' USING FORMATR-PTR.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.

Default State

On

USRVALON

UserTypeValidationIsOn() returns the current state of user-defined type input field validation.

Working Storage Section

```
01  FORMATR-PTR             POINTER.
01  RETURN-CD               PIC S9(9) VALUE 0 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'USRVALON' USING FORMATR-PTR
                      RETURN-CD.
```

Usage Pointers	Input/ Output	Description
FORMATR-PTR	Input	Sends the address of the current NEONFormatter instance.
RETURN-CD	Output	Returns zero (0) if validation is turned off. Returns non-zero if validation is turned on.

OutMsg

The following is a COBOL wrapped version of the OutMsg class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

MSGOUTPT

GetMsgBuffer() and GetMsgLength() return the formatted output message and the output message length, using a pointer to hold the address of the formatted message.

Working Storage Section

```
01  OUT-MESSAGE-HOLD          POINTER.
01  MESSAGE-VALUE-OUT.
    05  MESSAGE-VAL-OUT       PIC X(80) VALUE SPACES.
        (PIC X(80) is an example and must be large enough to
         hold the output message)
01  MESSAGE-SIZE-OUT          PIC 9(9) VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'MSGOUTPT' USING OUT-MESSAGE-HOLD
                      MESSAGE-VALUE-OUT
                       MESSAGE-SIZE-OUT.
```

Usage Pointers	Input/ Output	Description
OUT-MESSAGE-HOLD	Input	Sends a pointer that holds the address of the formatted message.
MESSAGE-VALUE-OUT	Output	Returns the output formatted message to be written to an output file.
MESSAGE-SIZE-OUT	Output	Returns the output message length.

OutMsgGroup

The following is a COBOL wrapped version of the OutMsgGroup class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

MSGCOUNT

GetMsgCount() returns the number of output messages in an output message group, using the message counter, a pointer to the out-message group, and a pointer that holds the address of the output message.

Working Storage Section

```
01  MSG-CNT                  PIC S9(9) VALUE 0 USAGE IS BINARY.
01  OUT-MESSAGE-PTR          POINTER.
01  OUT-MESSAGE-HOLD         POINTER.
```

COBOL calling portion

```
CALL 'MSGCOUNT' USING MSG-CNT
                      OUT-MESSAGE-PTR
                     OUT-MESSAGE-HOLD.
```

Usage Pointers	Input/ Output	Description
MSG-CNT	Output	Returns the number of messages.
OUT-MESSAGE-PTR	Input	Sends a pointer that holds the address of the out message group.
OUT-MESSAGE-HOLD	Input	Sends a pointer that holds the address of the formatted message.

ParsedField

The following is a list of the COBOL wrapped version of the ParsedField class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

GTFLDNM

Get Field Name returns the name of the field in a parsed message.

Working Storage Section

```
01  PARSEFLD-PTR              POINTER.
01  OUT-FLDNAME-PTR           PIC X(33) VALUE SPACES.
```

COBOL calling portion

```
CALL 'GTFLDNM' USING PARSEFLD-PTR
                     OUT-FLDNAME-PTR.
```

Usage Pointers	Input/ Output	Description
PARSEFLD-PTR	Input	Sends the parsed field class.
OUT-FLDNAME-PTR	Output	Returns the name of the field.

GTASCVAL

GetAsciiValue() returns the string value of the specified field in a parsed message.

Working Storage Section

```
01  PARSEFLD-PTR            POINTER.
01  ASCII-IDX               PIC 9(9) VALUE 00 USAGE IS BINARY.
01  GET-ASCII-VALUE.
    05  ASCII-VAL           PIC X(19624) VALUE SPACES.
01  VAL-LENGTH              PIC 9(9) VALUE 00 USAGE IS BINARY.
01  GET-ASCII-VALUE-TABLE.
    05  ASCII-VAL-TABLE     PIC X(01)
              OCCURS 1 TO 19624 TIMES
                 DEPENDING ON VAL-LEN.
```

COBOL calling portion

```
CALL 'GTASCVAL' USING PARSEFLD-PTR
                      ASCII-VAL
                      ASCII-IDX.
MOVE ASCII-IDX TO VAL-LEN.
MOVE ASCII-VAL TO GET-ASCII-VALUE-TABLE.
```

Usage Pointers	Input/ Output	Description
PARSEFLD-PTR	Input	Sends the parsed field class.
ASCII-VAL	Output	Returns the ASCII value of the parsed field.
ASCII-IDX	Output	Returns the data length.

GTFLDVAL

GetValue() returns the value of the specified field in a parsed message in its original data type.

Working Storage Section

```
01  PARSEFLD-PTR            POINTER.
01  DATA-TYPE               PIC 9(9) VALUE 00 USAGE IS BINARY.
01  FLD-LENGTH              PIC 9(9) VALUE 00 USAGE IS BINARY.
01  GET-FIELD-VALUE.
    05  GET-FIELD-VAL       PIC X(19624) VALUE SPACES.
01  FIELD-VALUE-TABLE.
    05  FIELD-VAL-TABLE     PIC X(01)
              OCCURS 1 TO 19624 TIMES
                 DEPENDING ON FLD-LEN.
```

COBOL calling portion

```
CALL 'GTFLDVAL' USING PARSEFLD-PTR
                      DATA-TYPE
                     FLD-LENGTH
                   GET-FIELD-VAL.
MOVE FLD-LENGTH TO FLD-LEN.
MOVE GET-FIELD-VAL TO FIELD-VALUE-TABLE.
```

Usage Pointers	Input/ Output	Description
PARSEFLD-PTR	Input	Sends the parsed field class.
DATA-TYPE	Output	Returns the data type.
FLD-LENGTH	Output	Returns the length of the data.
GET-FIELD-VALUE	Output	Returns the value of the parsed field in its original data type.

ParsedMessage

The following is a list of the COBOL wrapped version of the ParsedMessage class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

GTCMPCNT

GetCompCount() returns the number of components (messages or fields) in a parsed message.

Working Storage Section

```
01  PARSEMSG-PTR           POINTER.
01  COMP-COUNT             PIC 9(9) VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'GTCMPCNT' USING PARSEMSG-PTR
                      COMP-COUNT.
```

Usage Pointers	Input/ Output	Description
PARSEMSG-PTR	Input	Sends a pointer to the parsed message class.
COMP-COUNT	Output	Returns the number of components in a parsed message.

GTMSGCMP

GetMsgComp() returns the message component at the specified index. It returns NULL if a bad index is supplied or a flat format is used.

Working Storage Section

```
01  PARSEMSG-PTR            POINTER.
01  PARSEMESG-PTR           POINTER VALUE NULL.
01  MSGCOMP-IDX             PIC 9(9) VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'GTMSGCMP' USING PARSEMSG-PTR
                      PARSEMESG-PTR
                        MSGCOMP-IDX.
```

Usage Pointers	Input/ Output	Description
PARSEMSG-PTR	Input	Sends the parsed input message.
PARSEMESG-PTR	Output	Returns the message component at the specified index.
MSGCOMP-IDX	Input	Sends the index of the field to return.

GETINFO

GetInfo() returns the format name and type of the parsed message.

Working Storage Section

```
01  PARSEMSG-PTR            POINTER.
01  IN-FMTNAME-PTR          PIC X(33) VALUE SPACES.
01  INFO-IDX                PIC 9(9) VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'GETINFO' USING PARSEMSG-PTR
                     IN-FMTNAME-PTR
                           INFO-IDX.
```

Usage Pointers	Input/ Output	Description
PARSEMSG-PTR	Input	Sends the parsed input message.
IN-FMTNAME-PTR	Output	Returns the format name of the parsed message.
INFO-IDX	Input	Sends the type of the format. 1 = FLAT FORMAT 2 = COMPOUND FORMAT

GTFLDCMP

GetFieldComp() returns the component field at the specified index. It returns NULL if a bad index is supplied or a compound_format is used.

Working Storage Section

```
01  PARSEMSG-PTR          POINTER.
01  PARSEFLD-PTR          POINTER VALUE NULL.
01  FLD-IDX               PIC 9(9) VALUE 00 USAGE IS BINARY.
```

COBOL calling portion

```
CALL 'GTFLDCMP' USING PARSEMSG-PTR
                      PARSEFLD-PTR
                           FLD-IDX.
```

Usage Pointers	Input/ Output	Description
PARSEMSG-PTR	Input	Sends the parsed input message.
PARSEFLD-PTR	Output	Returns the field at the index specified.
FLD-IDX	Input	Sends the index of the field to return.

Chapter 5

Testing Rules

This chapter describes how to test NEONRules using a COBOL driver program. It provides a summary of the API calls, a sample source code flow chart of the main calls, and detailed API descriptions.

RULTSTCB

RULTSTCB is the COBOL test program. Its components are.

- a COBOL driver program
- API wrappers called by the COBOL program that interface between the COBOL program and the C++ modules
- C++ modules that exercise NEONRules APIs.

Notes:

For the complete sample program, see *RULTSTCB Sample Program* on page 110.

For sample JCL to run program RULTSTCB, see *Running the COBOL Test Programs* on page 89.

For more information on NEONRules, see the ***System Management Guide***.

RULTSTCB API Calls

The following list summarizes the flow of API calls in the order in which they are used in RULTSTCB.

STARTSES begins a DBMS or database session.

```
CALL 'STARTSES' USING SESSION-PTR
                      SESSION-NAME
                          DB2-TYPE.
```

CRRULENG creates the NEONRules daemon.

```
CALL 'CRRULENG' USING SESSION-PTR
                          RULE-PTR.
```

LRULSET loads the rules.

```
CALL 'LRULESET' USING RULE-PTR
                       APP-NAME
                      MESS-NAME
                       LOAD-NOW
                      RTRN-CODE.
```

EVAL retrieves the rules, parses the message, and evaluates the message based on evaluation criteria.

```
CALL 'EVAL' USING RULE-PTR
                    APP-NAME
                   MESS-NAME
                INPUT-RECORD
                  IN-REC-LEN.
```

GETRULE loads the rules into hit and no-hit tables.

```
CALL 'GETRULE' USING RULE-PTR
                       HIT-COUNT
                    NO-HIT-COUNT
                  HIT-RULE-TABLE
              NO-HIT-RULE-TABLE.
```

LOADTAB retrieves subscriptions, actions, options, and values.

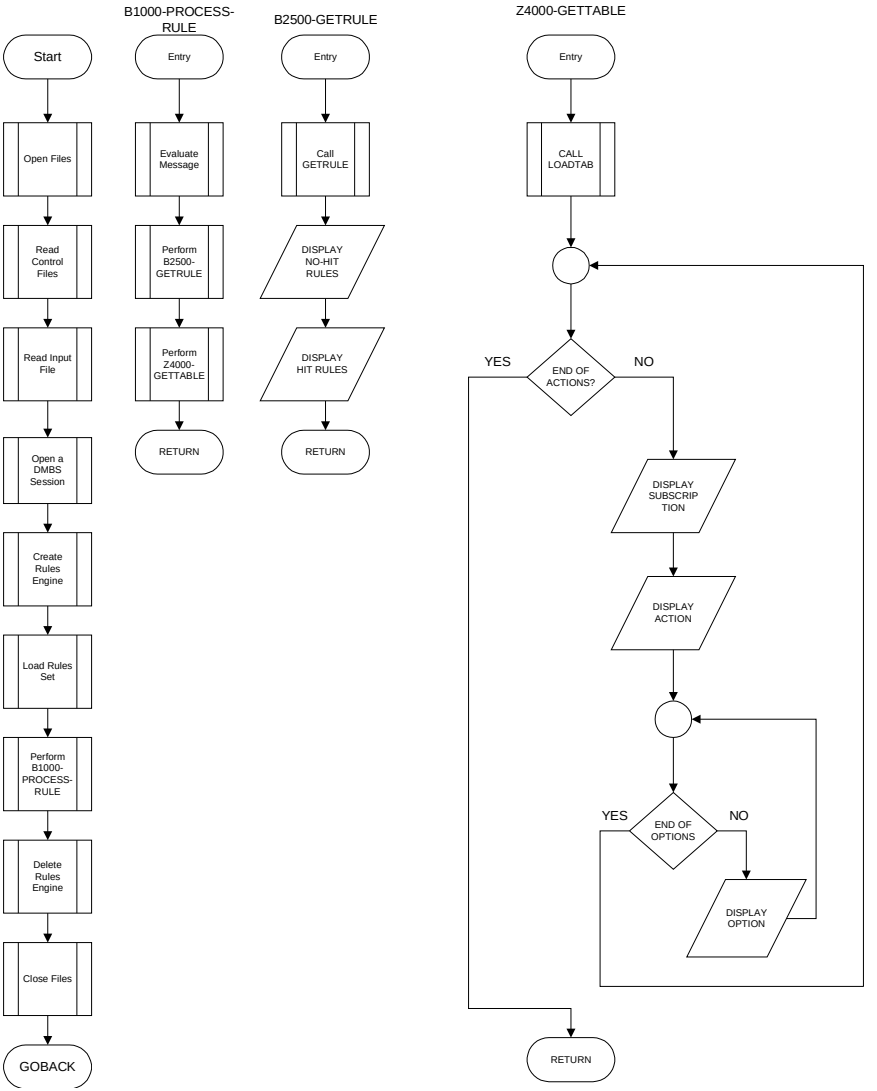
```
CALL 'LOADTAB' USING RULE-PTR
                     ACTION-CNTR
                     OPTION-CNTR
            OPTION-COUNT-TABLE
```

```
                ACTION-ARRAY
                   SUB-ARRAY
                OPTION-ARRAY.
```

DLRULENG deletes the NEONRules daemon.

```
CALL 'DLRULENG' USING RULE-PTR.
```

RULTSTCB Flow of Calls



RULTSTCB Example Output Display

The following is a sample output display of the RULTSTCB program. It illustrates the number of rules hit, the names of the hit rules, and the actions that would be performed.

```
NUMBER OF NO HIT RULES IS 0000001

NUMBER OF HIT RULES IS 0000001
------NO HIT RULES------
CCinNRule
-------HIT RULES-------
CCinRule

IN Z4000-GETTABLE
RULE-PTR IS 0203807064
THERE ARE 0000003 ACTIONS
THERE ARE 0000006 OPTIONS

SUBSCRIPTION           ACTION           OPTION
000000002
                      reformat
                                          INPUT_FORMAT:CCIN
                                       TARGET_FORMAT:CCOUT
```

API Reference for COBOL Wrappers

This section details the COBOL wrapped version of the NEONRules APIs that are contained in the sample RULTSTCB program. The wrappers are listed by class.

For more information on NEONRules APIs, see ***Programming Reference for NEONRules***, p. 33.

DbmsSession

The following is a COBOL wrapped version of the DbmsSession class C++ APIs. The associated C++ API name is in parentheses following the COBOL name..

STARTSES

OpenDbmsSession() creates the Dbms session.

Working Storage Section

```
01  SESSION-PTR              POINTER    VALUE NULL.
01  SESSION-NAME             PIC X(08)  VALUE 'SESSION-NAME'.
01  DATABASE-VALUES.
      05  DB2-TYPE           PIC S9(9)  VALUE 5 USAGE IS BINARY.
```

COBOL Calling Portion

```
CALL 'STARTSES' USING SESSION-PTR
                      SESSION-NAME
                          DB2-TYPE.
```

Usage Pointers	Input/ Output	Description
SESSION-PTR	Output	Returns the address of the session.

Usage Pointers	Input/ Output	Description
SESSION-NAME	Input	Sends the name of DBMS session-defined in SQLSVSES file.
DB2-TYPE	Input	Sends the type of database used.

VRule

The following is a COBOL wrapped version of the VRule class C++ APIs. The associated C++ API name is in parentheses following the COBOL name...

CRRULENG

CreateRulesEngine() creates the NEONRules daemon by calling the wrapper and sending it the session pointer and a NULL NEONRules pointer.

Working Storage Section

```
01  SESSION-PTR            POINTER.
01  RULE-PTR               POINTER   VALUE NULL.
```

COBOL Calling Portion

```
CALL 'CRRULENG' USING SESSION-PTR
                      RULE-PTR.
```

Usage Pointers	Input/ Output	Description
SESSION-PTR	Input	Sends the address of the current session.
RULE-PTR	Output	Returns the address of the NEONRules daemon.

DLRULENG

DeleteRuleEngine() specifies which daemon to delete, using the NEONRules pointer.

Working Storage Section

```
01  RULE-PTR                POINTER.
```

COBOL Calling Portion

```
CALL DLRULENG USINGRULE-PTR
```

Usage Pointers	Input/ Output	Description
RULE-PTR	Input	Sends a pointer to specify which daemon to delete.

EVAL

Using the application group and the message type to retrieve the associated actual rules, eval() parses the input message into fields and evaluates those fields based on evaluation criteria retrieved from the NEONRules database.

Use GETRULE and LOADTAB to view evaluation results.

Working Storage Section

```
01  RULE-PTR               POINTER.
01  APP-GROUP              PIC X(33) VALUE 'APPLICATION-GROUP-
                           NAME'.
01  MESSAGE TYPE           PIC X(33) VALUE 'MESSAGE-TYPE-NAME'.
01  INPUT-RECORD.
    05 INPUT-REC           PIC X(01)
                           OCCURS 1 TO 19624
                           DEPENDING ON IN-REC-LEN
                           INDEXED BY IN-IDX.
01  ACCUMULATORS.
    05 IN-REC-LEN          PIC 9(09).
```

COBOL Calling Portion

```
CALL EVAL USING RULE-PTR
                APP-GROUP
             MESSAGE TYPE
             INPUT-RECORD
               IN-REC-LEN.
```

Usage Pointers	Input/ Output	Description
RULE-PTR	Input	Sends the address of the current NEONRules daemon.
APP-GROUP	Input	Sends the address of the application name.
MESSAGE TYPE	Input	Sends the address of the message name.
INPUT-RECORD	Input	Sends the address of the message.

Usage Pointers	Input/ Output	Description
IN-REC-LEN	Input	Sends the size of the message buffer.

GETRULE

gethitrule() and getnohitrule() retrieve the hit and no-hit rules for the current rule session.

Working Storage Section

```
01  RULE-PTR               POINTER.
01  HIT-COUNT              PIC 9(07)   VALUE 1 USAGE IS BINARY.
01  NO-HIT-COUNT           PIC 9(07)   VALUE 1 USAGE IS BINARY.
01  HIT-RULE-TABLE.
    05  HIT-RULELIST       PIC X(33)
                           OCCURS 1 TO 20000 TIMES
                           DEPENDING ON HIT-COUNT.
01  NO-HIT-RULE-TABLE.
    05  NO-HIT-RULELIST    PIC X(33)
                           OCCURS 1 TO 20000 TIMES
                           DEPENDING ON NO-HIT-COUNT.
```

COBOL Calling Portion

```
CALL 'GETRULE' USING RULE-PTR
                     HIT-COUNT
                  NO-HIT-COUNT
               HIT-RULE-TABLE
            NO-HIT-RULE-TABLE.
```

Usage Pointers	Input/ Output	Description
RULE-PTR	Input	Sends the address of the current NEONRules daemon.
HIT-RULE-TABLE	Output	Returns a table populated with hit rules.
NO-HIT-RULE-TABLE	Output	Returns a table populated with no-hit rule.
HIT-COUNT	Output	Returns the number of hit rules.
NO-HIT-COUNT	Output	Returns the number of no-hit rules.

WARNING!
To avoid corrupting the register when calling APIs, use subscripting by data-name instead of indexing for COBOL tables. For more information, see IBM COBOL for MVS&VM Language Reference (SC26-4769).

Return Value

Returns tables with the number of hit rules and no-hit rules.

LOADTAB

getsubscription(), getaction(), and getopt() retrieves the subscriptions, action, and options associated with a rule that evaluated to true.

Working Storage Section

```
01  RULE-PTR                POINTER.
01  ACTION-CNTR             PIC 9(07)  VALUE 0 USAGE IS BINARY.
01  OPTION-CNTR             PIC 9(07)  VALUE 0 USAGE IS BINARY.
01  OPTION-COUNT-TABLE.
    05  OPT-COUNT-LIST      PIC 9(09)  VALUE 0 USAGE IS BINARY.
                            OCCURS 1 TO 20 TIMES
                            DEPENDING ON ACTION-CNTR.
01  ACTION-ARRAY.
    05 ACTION-X             PIC X(33)
                            OCCURS 1 TO 20 TIMES
                            DEPENDING ON ACTION-CNTR.
01  SUB-ARRAY.
    05 SUB-X                PIC 9(09)  VALUE 1 USAGE IS BINARY
                            OCCURS 1 TO 20 TIMES
                            DEPENDING ON ACTION-CNTR.
01  OPTION-ARRAY.
    10 TOT-OPTION           PIC X(66)
                            OCCURS 1 TO 20 TIMES
                            DEPENDING ON OPTION-CNTR.
```

COBOL Calling Portion

```
CALL 'LOADTAB' USING RULE-PTR
                     ACTION-CNTR
                     OPTION-CNTR
               OPTION-COUNT-TABLE
                   ACTION-ARRAY
                      SUB-ARRAY
                   OPTION-ARRAY.
```

Usage Pointers	Input/ Output	Description
RULE-PTR	Input	Sends the address of the current NEONRules daemon.
ACTION-CNTR	Output	Returns the number of actions that occurred.
OPTION-CNTR	Output	Returns the number of options that occurred.
OPTION-COUNT-TABLE	Output	Returns a list of occurrences for each action.
ACTION-ARRAY	Output	Returns a table populated with actions.
SUB-ARRAY	Output	Returns a table populated with subscriptions.
OPTION-ARRAY	Output	Returns a table populated with options.

WARNING!
To avoid corrupting the register when calling APIs, use subscripting by data-name instead of indexing for COBOL tables. For more information, see IBM COBOL for MVS&VM Language Reference (SC26-4769).

Return Value

Returns tables with the number of retrieved actions, subscriptions, and options.

LoadRuleSet

The following is a list of the COBOL wrapped version of the LoadRuleSet class C++ APIs. The associated C++ API name is in parentheses following the COBOL name.

LRULESET

LoadRuleSet() loads the rule set indicated by the application group and the message type.

Working Storage Section

```
01  RULE-PTR                POINTER.
01  APP-NAME                PIC X(33) VALUE 'APPLICATION-GROUP-
                             NAME'.
01  MESS-NAME               PIC X(33) VALUE 'MESSAGE-TYPE-
                             NAME'.
01  LOAD-NOW                PIC 9(07) VALUE 0 USAGE IS BINARY.
01  RETURN-CODE             PIC 9(07) VALUE 0 USAGE IS BINARY.
```

COBOL Calling Portion

```
CALL 'LRULESET' USING RULE-PTR
                      APP-NAME
                     MESS-NAME
                       LOAD-NOW
                    RETURN-CODE.
```

Usage Pointers	Input/ Output	Description
RULE-PTR	Input	Sends the address of the current NEONRules daemon.
APP-NAME	Input	Sends the address of the application name.
MESS-NAME	Input	Sends the address of the message name.

Usage Pointers	Input/ Output	Description
LOAD-NOW	Input	Sends a message when to reload NEONRules Set information. 0 = (default) Reloads rule set information when next EVAL is called. 1 = Reloads immediately.
RETURN-CODE	Output	Returns the return code of Load Rule Set.

Return Value

Returns 1 if the reload indicator was set for the rule set indicated and the load was performed. Returns 2 if the reload indicator was set for the rule set indicated but the rule set was not loaded. Returnszero (0) if the load was not be performed.

If an error is encountered, use GetErrorNo() to retrieve the error number and GetErrorMessage() to retrieve the error message associated with the error.

Internal Functions

The following functions are handled internally:

- GETLOG
- GETRRNO
- GTERRMSG
- GETRERR

If an error is encountered, the error message displays in the SYSPRINT.

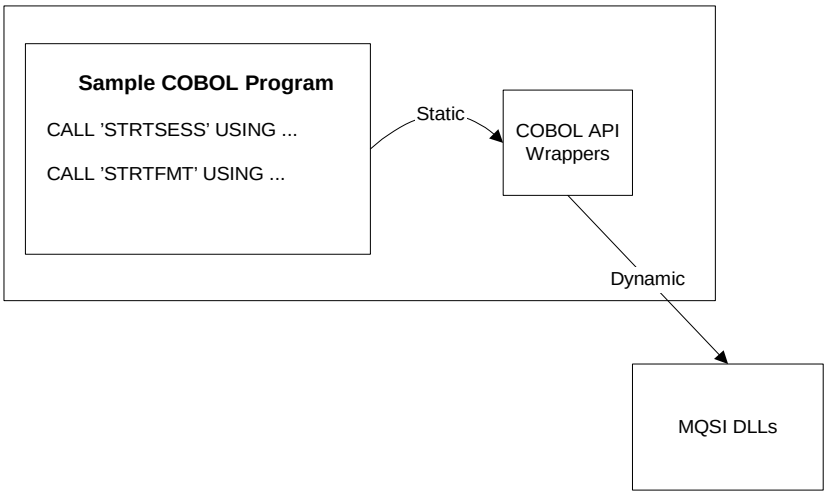
Chapter 6

Compiling the Sample COBOL Programs

This chapter illustrates how a COBOL application calls MQSeries Integrator APIs, provides procedures for compiling and linking the sample COBOL programs and for running and troubleshooting the COBOL test programs.

Calling MQSeries Integrator APIs

The following diagram illustrates the interrelationship of the COBOL program, COBOL API wrappers and the MQSI DLLs when a COBOL application calls the MQSeries Integrator APIs. The load modules for each wrapper are statically linked to the COBOL program; however, communication with the MQSI DLLs containing the C++ library functions is dynamically established.



Compiling and Linking

To compile and link the sample programs in your environment, you must modify the OS/390 jobs as illustrated in the SNEOJCL library. This includes creating a valid JOBCARD for your site and valid dataset names for IBM Language Environment and COBOL runtime libraries.

1. Replace the substitution string in the compile JCL in the <hlq>.SNEOJCL library with the appropriate libraries for your site.
   - MSGTSTCC - compiles MSGTSTCB
   - MSGTSTCB - executes MSGTSTCB
   - RULTSTCC - compiles RULTSTCB
   - RULTSTCB - executes RULTSTCB
   - TESTCOBC - compiles TESTCOB1

   Note:
   Source programs can be found in the <hlq>.SNEOSRCE library.

2. Replace the substitution string in the COBCOMP PROC in the <hlq>.SNEOPROC library with the appropriate libraries for your site. To link the wrappers to the programs when the COBOL programs MSGTSTCB, RULTSTCB, and TESTCOB1 are compiled, the load library for <hlq>.SNEOLKED must be in the SYSLIB list in the LinkEdit step of the COBCOMP PROC. The following sample illustrates the LinkEdit step:

   ```
   LKED

   SYSLIB  DD  DSN=<hlq>.SNEOLKED,DISP=SHR
   ```

3. Run the jobs to compile the programs. Because an executable module for each sample program is distributed in the <hlq>.SNEOLOAD library, place your compiled version in a separate load module for testing.
4. Define the DB2 plan before running the sample programs. For more information, see the ***Installation and Configuration Guide***.

Tailoring Sample JCL

Compile PROC

The following sample JCL shows how to compile the COBOL/COBOL II sample programs.

```
//*************************************************************
//*                                                           *
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.        *
//* All Rights Reserved.                                      *
//*                                                           *
//* Release 4.1.1                                             *
//*************************************************************
//*************************************************************
//*
//*  COMPILE PRELINK AND LINK A COBOL PROGRAM
//*
//*  OS/390 COBOL
//*
//*  RELEASE LEVEL: 02.04.00  (VERSION.RELEASE.MODIFICATION
//*  LEVEL)
//*
//*************************************************************
//*
//COBCOMP PROC INFILE=,                 < INPUT ... REQUIRED
//   OUTFILE=,                          < TARGET SYSLMOD
//   MEMBER=,                           < SOURCE MEMBER NAME
//   CPARM='RENT,NUM,XREF,LIST,MAP',
//   COBPRFX='IGY.V1R2M0',SYSLBLK=3200,
//   LIBPRFX='CEE',                     < PREFIX FOR LIBRARY
//                                        DSN
//   SMPHLQ='<smphlq>',                 < PREFIX FOR
//                                        DEVELOPMENT DSN
//   LPARM='AMODE=31,MAP,RENT,XREF,LIST,LET',  <TOR OPTIONS
//   TUNIT='SYSALLDA'                   <UNIT FOR TEMPORARY ///
//                                       FILES
//   MQSDSN='<mqshlq>.SCSQLOAD',        < MQSERIES LOAD LIBRARY
//   DB2DSN='<db2hlq>.SDSNLOAD'         < DB2 LOAD LIBRARY
//*-------------------------------------------------------------
//*  COMPILE STEP:
//*-------------------------------------------------------------
//*
//COMPILE EXEC PGM=IGYCRCTL,REGION=2048K,
//           PARM='&CPARM'
//STEPLIB  DD  DSNAME=&COBPRFX..SIGYCOMP,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=&&LOADSET,UNIT=SYSDA,
```

```
//             DISP=(MOD,PASS),SPACE=(TRK,(3,3)),
//             DCB=(BLKSIZE=&SYSLBLK)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  DSN=&INFILE(&MEMBER),DISP=SHR
//*
//*-------------------------------------------------------------
//* LINKEDIT STEP:
//*-------------------------------------------------------------
//LKED   EXEC PGM=HEWL,REGION=1024K,COND=(8,LT,COMPILE),
//       PARM='&LPARM'
//SYSLIB   DD  DSN=&LIBPRFX..SCEELKED,DISP=SHR
//************************************************************
//**The loadlib that houses the API wrappers and aliases ******
//**must be in the syslib concatenation **********************
//************************************************************
//         DD  DSN=&SMPHLQ..SNEOLKED,DISP=SHR
//         DD  DSN=&SMPHLQ..SNEODLL,DISP=SHR
//SYSOBJ   DD  DSN=&&OBJLIB,UNIT=&TUNIT.,DISP=(NEW,PASS),
//             SPACE=(32000,(30,30)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIN   DD  DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DSN=&OUTFILE(&MEMBER),DISP=SHR
//MQSLOAD  DD  DSN=&MQSDSN,DISP=SHR
//DB2LOAD  DD  DSN=&DB2DSN,DISP=SHR
//SYSUT1   DD  UNIT=&TUNIT.,SPACE=(32000,(30,30))
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  DUMMY
//*
```

MSGTSTCC

The following sample JCL shows how to compile MSGTSTCB.

```
//* <insert a valid JOBCARD for your site>
//************************************************************
//*                                                          *
```

```
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.        *
//* All Rights Reserved.                                      *
//*                                                           *
//* Release 4.1.1                                             *
//************************************************************
//PROCLIST JCLLIB
 ORDER=(<smphlq>.SNEOPROC,SYS1.PROCLIB,SYS2.PROCLIB)
//*
//*
//COMPILE  EXEC COBCOMP,LIBPRFX=CEE,MEMBER=MSGTSTCB,
//        OUTFILE='<smphlq>.SNEOLOAD'
//COMPILE.SYSIN DD DSN=<smphlq>.SNEOSRCE(MSGTSTCB),DISP=SHR
//LKED.SYSLMOD DD UNIT=SYSALLDA
```

RULTSTCC

The following sample JCL shows how to compile RULTSTCB.

```
//* <insert a valid JOBCARD for your site>
//************************************************************
//*                                                           *
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.        *
//* All Rights Reserved.                                      *
//*                                                           *
//* Release 4.1.1                                             *
//************************************************************
//PROCLIST JCLLIB
 ORDER=(<smphlq>.SNEOPROC,SYS1.PROCLIB,SYS2.PROCLIB)
//*
//COMPILE  EXEC COBCOMP,LIBPRFX=CEE,MEMBER=RULTSTCB,
//        OUTFILE='<smphlq>.SNEOLOAD'
//COMPILE.SYSIN DD DSN=<smphlq>.SNEOSRCE(RULTSTCB),DISP=SHR
//LKED.SYSLMOD DD UNIT=SYSALLDA
//LKED.SYSIN   DD *
  NAME    RULTSTCB(R)
```

Running the COBOL Test Programs

The MSGTSTCB, RULTSTCB, and TESTCOB1 programs are designed to test the COBOL API functions for NEONFormatter and NEONRules.

MSGTSTCB

The following sample JCL shows how to execute MSGTSTCB. This program reads one record from an input file, parses the input message using the input format and displays the field name and value in SYSOUT. The output format is used to reformat the message, which is written to a file.

For more information on testing formats, See *Testing Formats* on page 31.

For a complete MSGTSTCB sample program, See *MSGTSTCB Sample Program* on page 95.

```
//<insert a valid jobcard for your site>
//*************************************************************
//*                                                           *
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.        *
//* All Rights Reserved.                                      *
//*                                                           *
//* Release 4.1.1                                             *
//*************************************************************
//*************************************************************//
//*                                                           *//
//* MSGTSTCB: Tests the COBOL version of MSGTEST              *//
//*                                                           *//
//*************************************************************//
//MSGTSTCB PROC SMPHLQ='<smphlq>',      HLQ for NEON distrib libs
//         MQSHLQ='<mqshlq>',      HLQ for MQS runtime libs
//         CEEHLQ='<ceehlq>',      HLQ for Lang Envir  libs
//         CSSHLQ='SYS1',          HLQ for Callable Sys Svcs (CSS-)Lib
//         SQLMEM='SQLSVSES',      MEMbername for SQLSVSES cntl cards
//         OPCLAS='*'              SYSOUT CLASS
//*
//*
//STP0101 EXEC PGM=MSGTSTCB
```

```
//*
//<customize member STEPLIB and copy at this point in the JCL>
//*
//CEEDUMP  DD  SYSOUT=&OPCLAS
//SYSUDUMP DD  SYSOUT=&OPCLAS
//SYSOUT   DD  SYSOUT=&OPCLAS
//SYSPRINT DD  SYSOUT=&OPCLAS
//SQLSVSES DD  DSN=&SMPHLQ..SNEOCNTL(&SQLMEM),DISP=SHR
//FRMTFILE DD  DSN=&SMPHLQ..SNEOCNTL(FMTNAME),DISP=SHR
//SESSFILE DD  DSN=&SMPHLQ..SNEOCNTL(SESSNAME),DISP=SHR
//         PEND
//MSGTSTCB EXEC MSGTSTCB
//INPUTT   DD  DISP=SHR,DSN=<your-input-file>
//OUTFILE  DD  DISP=SHR,DSN=<your-output-file>
//
/*
```

Troubleshooting MSGTSTCB

- If you are unable to resolve the wrappers when compiling, verify that the <hlq>.SNEOLKED library is in the SYSLIB concatenation of the LINKEDIT step in the compile PROC.
- If the static wrapper and the application cannot find the correct DLL, verify that <hlq>.SNEODLL library is in the STEPLIB concatenation of the execution JCL.
- If the message output is blank when displaying a reformatted message, verify that the table into which the message is moved is initialized with the length variable.
- If the error "mandatory input field fieldname not found" is received, verify that the input message file is correct for the formats used.

RULTSTCB

The following sample JCL shows how to execute RULTSTCB. This program reads one record from an input file. Using the application group and message type, RULTSTCC evaluates the message. This program shows rules that are hit and no-hit. The actions and options for each hit rule will be displayed.

For more information on testing rules, see See *Testing Rules* on page 65.

For a complete RULTSTCB sample program, See *RULTSTCB Sample Program* on page 110.

```
//<insert a valid jobcard for your site>
//************************************************************
//*                                                          *
//* Licensed Materials - Property of New Era of Networks, Inc.*
//* Copyright (c) 1998-1999, New Era of Networks, Inc.       *
//* All Rights Reserved.                                     *
//* Release 4.1.1                                            *
//************************************************************
//*****************************************************************/
//*                                                               */
//* RULTSTCB: Tests the COBOL version of RULTEST                  */
//*                                                               */
//*****************************************************************/
//RULTSTCB PROC SMPHLQ='<smphlq>',      HLQ for NEON distrib libs
//         MQSHLQ='<mqshlq>',      HLQ for MQS runtime libs
//         CEEHLQ='<ceehlq>',      HLQ for Lang Envir  libs
//         CSSHLQ='SYS1',          HLQ for Callable Sys Svcs (CSS-)Lib
//         SQLMEM='SQLSVSES',      MEMbername for SQLSVSES cntl cards
//         OPCLAS='*'              SYSOUT CLASS
//*
//*
//STP0101 EXEC PGM=RULTSTCB
//*
//<customize member STEPLIB and copy at this point in the JCL>
//*
//CEEDUMP  DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//ERRFILE  DD  DSN=<Your error log file here>,DISP=SHR
//SQLSVSES DD  DSN=&SMPHLQ..SNEOSNTL(&SQLMEM),DISP=SHR
```

```
//APPFILE  DD  DSN=&SMPHLQ..SNEOCNTL(APPNAME),DISP=SHR
//FRMTFILE DD  DSN=&SMPHLQ..SNEOCNTL(FMTNAME),DISP=SHR
//SESSFILE DD  DSN=&SMPHLQ..SNEOCNTL(SESSNAME),DISP=SHR
//         PEND
//RULTSTCB EXEC RULTSTCB
//INPUTT   DD  DISP=SHR,DSN=<your-input-file>
//OUTFILE  DD  DISP=SHR,DSN=<your-output-file>
//
/*
```

Troubleshooting RULTSTCB

- If you are unable to resolve the wrappers when compiling, verify that the <hlq>.SNEOLKED library is in the SYSLIB concatenation of the LINKEDIT step in the compile PROC.
- If the static wrapper and the application cannot find the correct DLL, verify that <hlq>.SNEODLL library is in the STEPLIB concatenation of the execution JCL.
- RULTSTCB needs a minimum Region Size of 8M to run. If the program is started with a smaller Region Size, you may get messages like: *CEE3500S Not enough storage was available to load RULESFMT*, accompanied by a CEEDUMP.

Control Files for the COBOL Test Programs

Before running the COBOL test programs, you must edit members APPNAME, FMTNAME and SESSNAME in library <smphlq>.SNEOCNTL as follows.

APPNAME

Identifies an application group and message type. The two names should appear in the same record of this member, application group first, separated by one or more spaces. APPNAME is referenced only by RULTSTCB.

FMTNAME

Identifies an input format and an output ormat. The two names should appear in the same record of this member, input format first, separated by one or more spaces. FMTNAME is referenced only by MSGTSTCB.

SESSNAME

Identifies a session name that has been defined in member <smphlq>.SNEOCNTL(SQLSVSES). The name must contain no more than 31 characters. SESSNAME is referenced by both MSGTSTCB and RULTSTCB

Appendix A

COBOL Sample Programs

MSGTSTCB Sample Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MSGTSTCB.
AUTHOR. NEON - NEW ERA OF NETWORKS.

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.

*****************************************************************
* THIS PROGRAM IS A TEST DRIVER FOR THE FORMAT ENGINE WRAPPERS.
*
* THIS PROGRAM USES THE NEONET FORMATTER WRAPPERS:
*
* STRTSESS - ESTABLISH SESSION POINTER
* STRTFMTR - ESTABLISH A FORMAT POINTER, INSTANSTIATE FORMATTER
* ADNPTMSG - GET INPUT FORMAT FROM DATABASE AND LOAD INPUT
*            MESSAGE
* ADOPTFMT - GET OUTPUT FORMAT FROM DATABASE
* REFORMAT - REFORMAT THE INPUT MESSAGE USING THE INPUT FORMAT,
*            OUTPUT FORMAT AND INPUT MESSAGE
* PRSMSCNT - GET THE NUMBER OF PARSED MESSAGES
* GTPRSMSG - GET POINTER TO PARSED MESSAGE SPECIFIED BY INDEX
* GETINFO  - GET THE FORMAT NAME AND TYPE FOR THE PARSED MESSAGE
* GTCMPCNT - GETS THE NUMBER OF COMPONENTS (COMPNENTS OR FIELDS)
*            IN A PARSED MESSAGE
* GTMSGCMP - GET POINTER TO COMPONENT USING INDEX SPECIFIED
* GTFLDCMP - GET POINTER TO FIELD USING INDEX SPECIFIED
* GTFLDNM  - GET FIELD NAME
* GTFLDVAL - GET FIELD VALUE
```

```
* GETOTMSG - GET POINTER TO OUTPUT MESSAGES
* MSGCOUNT - GET THE NUMBER OF OUTPUT MESSAGES
* MSGOUTPT - GET THE OUTPUT MESSAGE AND MESSAGE LENGHT USING
*            THE SPECIFIED INDEX
*
* THIS PROGRAM WILL READ ONE RECORD FROM THE INPUT FILE AND
* TREAT IT AS ONE MESSAGE.
*
* TABLES (STACKS) ARE USED TO KEEP TRACK OF THE MESSAGE AND ALL
* SUB-MESSAGES (COMPONENTS).
*
************************************************************************
INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT INPUT-FILE
        ASSIGN TO INPUTT
        FILE STATUS IS INPUT-FILE-STATUS.

    SELECT OUTPUT-FILE
        ASSIGN TO OUTFILE
        FILE STATUS IS OUTPUT-FILE-STATUS.

    SELECT FORMAT-FILE
        ASSIGN TO FRMTFILE.

    SELECT SESSION-FILE
        ASSIGN TO SESSFILE.

DATA DIVISION.

************************************************************************
*
*  FILE LAYOUTS
*
************************************************************************
FILE SECTION.

FD  INPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS V
    RECORD IS VARYING FROM 1 TO 32752 CHARACTERS
```

```
                DEPENDING ON IN-REC-LEN
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS INPUT-RECORD.

01  INPUT-RECORD.
    05  INPUT-REC          PIC X(01)
                           OCCURS 1 TO 32752
                           DEPENDING ON IN-REC-LEN
                           INDEXED BY IN-IDX.

FD  FORMAT-FILE
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F.

01  FORMAT-RECORD          PIC X(80).

FD  SESSION-FILE
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F.

01  SESSION-RECORD         PIC X(80).

FD  OUTPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS V
    RECORD IS VARYING FROM 1 TO 32752 CHARACTERS
                      DEPENDING ON MESSAGE-SIZE-OUT
    LABEL RECORDS ARE STANDARD
    DATA RECORD IS OUTPUT-RECORD.

01  OUTPUT-RECORD.
    05  OUTPUT-REC         PIC X(01)
                           OCCURS 1 TO 32752 TIMES
                           DEPENDING ON MESSAGE-SIZE-OUT
                           INDEXED BY OUT-IDX.

WORKING-STORAGE SECTION.

*****************************************************************
*
```

```
*  PARAMETERS FOR NEONET
*
*****************************************************************
01  SESSION-PTR           POINTER   VALUE NULL.
01  FORMATR-PTR           POINTER   VALUE NULL.
01  PARSEFLD-PTR          POINTER   VALUE NULL.
01  OUT-MESSAGE-PTR       POINTER   VALUE NULL.
01  OUT-MESSAGE-HOLD      POINTER   VALUE NULL.
01  INPUT-MSG-PTR         POINTER   VALUE NULL.

01  POINTER-TABLE.
    05  SUB-MSG-PTR       POINTER   VALUE NULL
                          OCCURS 200 TIMES.
01  DB2-TYPE              PIC S9(09) VALUE 5 USAGE IS BINARY.
01  MESSAGE-SIZE-OUT      PIC  9(09) VALUE 0 USAGE IS BINARY.
01  DATA-TYPE             PIC  9(09) VALUE 0 USAGE IS BINARY.
01  DATA-LENGTH           PIC  9(09) VALUE 0 USAGE IS BINARY.
01  MSG-IDX               PIC  9(09) VALUE 0 USAGE IS BINARY.
01  FLD-IDX               PIC  9(09) VALUE 0 USAGE IS BINARY.
01  INFO-IDX              PIC  9(09) VALUE 0 USAGE IS BINARY.
    88  FLAT-FORMAT                  VALUE 1.
    88  COMPOUND-FORMAT              VALUE 2.
01  FLD-COUNT             PIC  9(09) VALUE 0 USAGE IS BINARY.
01  MSG-CNT               PIC S9(09) VALUE 0 USAGE IS BINARY.
01  PARSED-IN-NUM         PIC S9(09) VALUE 0 USAGE IS BINARY.

01  COMP-COUNT-TABLE.
    05  COMP-COUNT        PIC  9(09) VALUE 0 USAGE IS BINARY
                          OCCURS 200 TIMES.

01  MSGCOMP-IDX-TABLE.
    05  MSGCOMP-IDX       PIC  9(09) VALUE 0 USAGE IS BINARY
                          OCCURS 200 TIMES.

01  SESSION-NAME          PIC  X(33) VALUE SPACES.
01  IN-FMTNAME            PIC  X(32) VALUE SPACES.
01  OUT-FLDNAME           PIC  X(32) VALUE LOW-VALUES.

01  FORMAT-RECORD-FIELDS
    05  IN-FORMAT         PIC X(32) VALUE SPACES.
    05  OUT-FORMAT        PIC X(32) VALUE SPACES.
    05  FILLER            PIC X(16).
```

```
01  GET-FIELD-VALUE.
    05  GET-FIELD-VAL     PIC X(32767) VALUE LOW-VALUES.

01  MESSAGE-VALUE-OUT.
    05  MESSAGE-VAL-OUT   PIC X(32767) VALUE LOW-VALUES.

*****************************************************************
*
*  FLAGS
*
*****************************************************************
01  FLAGS.

    05 EOF-FLAG           PIC X  VALUE 'N'.
       88 EOF                    VALUE 'Y'.
*****************************************************************
*
*  WORK AREAS
*
*****************************************************************
01  INPUT-FILE-STATUS     PIC  X(02) VALUE SPACES.
01  OUTPUT-FILE-STATUS    PIC  X(02) VALUE SPACES.
01  PTR-SUB               PIC S9(04) COMP VALUE 0.
01  IN-REC-LEN            PIC  9(02) COMP VALUE O.
01  SESSION-LENGTH        PIC  9(02) COMP VALUE 0.
01  INPUT-FORMAT-LENGTH   PIC  9(02) COMP VALUE 0.
01  OUTPUT-FORMAT-LENGTH  PIC  9(02) COMP VALUE 0.
01  RETURN-CD             PIC S9(09) VALUE 0 USAGE IS BINARY.

01  WS-DISPLAY-FIELD.
    05  WS-FIELD-NAME     PIC  X(33) VALUE SPACES.
    05  FILLER            PIC  X(05) VALUE SPACES.
    05  WS-FIELD-VALUE    PIC  X(94) VALUE SPACES.

*****************************************************************
PROCEDURE DIVISION.

A0000-MAIN.
    PERFORM C1000-OPEN-FILES.

    IF INPUT-FILE-STATUS = ZEROES
```

```
       IF OUTPUT-FILE-STATUS = ZEROES

          PERFORM C2000-READ-CONTROL-FILES

          IF SESSION-LENGTH > 31
             DISPLAY 'Session name:  ' SESSION-RECORD
                     ' is too long. '
             DISPLAY
               'SESSION NAME MUST BE 31 BYTES OR LESS'
          ELSE
             IF INPUT-FORMAT-LENGTH = 0
                DISPLAY 'YOU MUST GIVE AN INPUT FORMAT NAME'
             ELSE
                IF OUTPUT-FORMAT-LENGTH = 0
                   DISPLAY 'YOU MUST GIVE AN OUTPUT FORMAT NAME'
                ELSE
                   PERFORM C3000-READ-INPUT-FILE.
                   PERFORM A1000-PARSE-MESSAGE.
                   PERFORM D1000-CLOSE-FILES.
                END-IF
             END-IF
          END-IF

       ELSE
          DISPLAY 'ERROR OPENING OUTPUT FILE.  FILE STATUS:  '
                  OUTPUT-FILE-STATUS.
       END-IF
    ELSE
       DISPLAY 'ERROR OPENING INPUT FILE.  FILE STATUS:  '
               INPUT-FILE-STATUS
    END-IF.

    GOBACK.

*****************************************************************
* A1000-PARSE-MESSAGE.
*
* ESTABLISH SESSION POINTER.
*
* INSTANTIATE THE FORMATTER.
*
* GET THE INPUT AND OUTPUT FORMATS AND REFORMAT THE INPUT
```

```
* MESSAGE.
*
* IF THE REFORMAT IS SUCCESSFUL, PARSE THE INPUT MESSAGE TO
* GET ALL THE FIELD NAMES AND VALUES.
*
* WRITE THE OUTPUT MESSAGE TO A FILE.
*
*****************************************************************
A1000-PARSE-MESSAGE.
    CALL 'STRTSESS' USING SESSION-PTR
                          SESSION-NAME
                          DB2-TYPE.

    IF SESSION-PTR = NULL
        DISPLAY 'ERROR CREATING SESSION'
    ELSE

        CALL 'STRTFMTR' USING SESSION-PTR
                              FORMATR-PTR

        CALL 'ADNPTMSG' USING FORMATR-PTR
                              IN-FORMAT
                              INPUT-RECORD
                              IN-REC-LEN

        CALL 'ADOPTFMT' USING FORMATR-PTR
                              OUT-FORMAT

        CALL 'REFORMAT' USING FORMATR-PTR
                              RETURN-CD

        IF RETURN-CD = 0

           DISPLAY 'ERROR IN REFORMAT'

        ELSE

            CALL 'PRSMSCNT' USING FORMATR-PTR
                                  PARSED-IN-NUM

            CALL 'GTPRSMSG' USING FORMATR-PTR
                                  MSG-IDX
```

```
                                INPUT-MSG-PTR

            CALL 'GETINFO' USING INPUT-MSG-PTR
                                 IN-FMTNAME
                                 INFO-IDX

            IF FLAT-FORMAT
                 PERFORM B5000-FLAT-FORMAT-ONLY

            ELSE
                IF COMPOUND-FORMAT
                    MOVE 1                    TO PTR-SUB
                    SET SUB-MSG-PTR (PTR-SUB) TO INPUT-MSG-PTR
                    PERFORM B2000-COMPOUND-FORMAT
                END-IF
            END-IF

            PERFORM B8000-GET-OUTPUT-MESSAGE

            DISPLAY 'REFORMAT COMPLETE'

        END-IF
    END-IF.

*****************************************************************
* B1000-TRAVERSE.
*
* DETERMINE THE TYPE OF FORMAT.
*
* FOR A FLAT FORMAT, GET THE FIELD NAMES AND VALUES FOR DISPLAY.
*
* FOR A COMPOUND FORMAT, USE THE NEXT SPACE IN THE STACK TO
* SAVE THE POINTER TO THE MESSAGE.
*
*****************************************************************

B1000-TRAVERSE.
    IF FLAT-FORMAT
       PERFORM B3000-GET-FLAT-FORMAT
    ELSE
        IF COMPOUND-FORMAT
            ADD 1   TO PTR-SUB
```

```
            PERFORM B2000-COMPOUND-FORMAT
        END-IF
    END-IF.

*****************************************************************
* B2000-COMPOUND-FORMAT.
*
* GET THE NUMBER OF COMPONENTS IN THE MESSAGE.
*
* LOOP THROUGH ALL THE COMPONENTS FOR THIS MESSAGE.
*
* SET THE SUBSCRIPT FOR THE STACK BACK THE PARENT MESSAGE.
*
*****************************************************************

B2000-COMPOUND-FORMAT.
    CALL 'GTCMPCNT' USING SUB-MSG-PTR (PTR-SUB)
                          COMP-COUNT  (PTR-SUB).

    MOVE 0                TO MSGCOMP-IDX (PTR-SUB).

    PERFORM B2500-GET-COMPOUND-FORMAT
        UNTIL MSGCOMP-IDX (PTR-SUB) = COMP-COUNT (PTR-SUB).

    SUBTRACT 1 FROM PTR-SUB.

    IF PTR-SUB < 0
       PERFORM B8000-GET-OUTPUT-MESSAGE

       DISPLAY 'REFORMAT COMPLETE'

       GOBACK

    END-IF.

*****************************************************************
* B2500-GET-COMPOUND-FORMAT.
*
* GET THE POINTER FOR THE COMPONENT SPECIFIED BY THE INDEX.
*
* INCREMENT THE INDEX.
*
```

```
* GET THE FORMAT NAME AND TYPE FOR THIS COMPONENT.
*
* DETERMINE WHAT TYPE OF FORMAT IT IS.
*
*****************************************************************

B2500-GET-COMPOUND-FORMAT.

    CALL 'GTMSGCMP' USING SUB-MSG-PTR (PTR-SUB)
                          SUB-MSG-PTR (PTR-SUB + 1)
                          MSGCOMP-IDX (PTR-SUB).

    ADD +1                TO MSGCOMP-IDX (PTR-SUB).

    CALL 'GETINFO' USING SUB-MSG-PTR (PTR-SUB + 1)
                          IN-FMTNAME
                          INFO-IDX.

    PERFORM B1000-TRAVERSE.

*****************************************************************
* B3000-GET-FLAT-FORMAT.
*
* GET THE NUMBER OF FIELDS IN THE MESSAGE.
*
* GET THE NAME AND VALUE FOR EACH FIELD.
*
*****************************************************************

B3000-GET-FLAT-FORMAT.

    CALL 'GTCMPCNT' USING SUB-MSG-PTR (PTR-SUB + 1)
                          FLD-COUNT.

    MOVE 0                TO FLD-IDX.

    PERFORM B4000-GET-FIELD
        UNTIL FLD-IDX = FLD-COUNT

*****************************************************************
* B4000-GET-FIELD.
*
```

```
* GTFLDCMP WILL RETURN A POINTER TO THE FIELD IN THE INPUT
* MESSAGE SPECIFIED BY THE FIELD INDEX.
*
* USE THE FIELD POINTER TO GET THE NAME AND VALUE OF THE FIELD.
*
*****************************************************************

B4000-GET-FIELD.
    CALL 'GTFLDCMP' USING SUB-MSG-PTR (PTR-SUB + 1)
                          PARSEFLD-PTR
                          FLD-IDX.

    CALL 'GTFLDNM' USING PARSEFLD-PTR
                          OUT-FLDNAME.

    CALL 'GTFLDVAL' USING PARSEFLD-PTR
                          DATA-TYPE
                          DATA-LENGTH
                          GET-FIELD-VAL.

   PERFORM B7000-DISPLAY-FIELD-INFO.

*****************************************************************
* B5000-FLAT-FORMAT-ONLY.
*
* THIS PARAGRAPH IS CALLED WHEN THE INPUT FORMAT IS A ONLY A
* FLAT FORMAT.  (ONLY ONE SET OF POINTERS IS USED IN THIS CASE.)
*
* GET THE NUMBER OF FIELDS IN THE MESSAGE.
*
* GET THE NAME AND VALUE FOR EACH FIELD.
*
*****************************************************************

B5000-FLAT-FORMAT-ONLY.

    CALL 'GTCMPCNT' USING INPUT-MSG-PTR
                          FLD-COUNT.

    MOVE 0                TO FLD-IDX.

    PERFORM B6000-GET-FIELD-FOR-FLAT-ONLY
```

```
        UNTIL FLD-IDX = FLD-COUNT.

*****************************************************************
* B6000-GET-FIELD-FOR-FLAT-ONLY.
*
* THIS PARAGRAPH IS CALLED WHEN THE INPUT FORMAT IS A ONLY A
* FLAT FORMAT.  (ONLY ONE SET OF POINTERS IS USED IN THIS CASE.)
*
* GTFLDCMP WILL RETURN A POINTER TO THE FIELD IN THE INPUT
* MESSAGE SPECIFIED BY THE FIELD INDEX.
*
* USE THE FIELD POINTER TO GET THE NAME AND VALUE OF THE FIELD.
*
*****************************************************************

B6000-GET-FIELD-FOR-FLAT-ONLY.

    CALL 'GTFLDCMP' USING INPUT-MSG-PTR
                          PARSEFLD-PTR
                          FLD-IDX.

    CALL 'GTFLDNM' USING PARSEFLD-PTR
                          OUT-FLDNAME.

    CALL 'GTFLDVAL' USING PARSEFLD-PTR
                          DATA-TYPE
                          DATA-LENGTH
                          GET-FIELD-VAL.

    PERFORM B7000-DISPLAY-FIELD-INFO.


*****************************************************************
* B7000-DISPLAY-FIELD-INFO.
*
* WRITE THE INPUT FIELD NAME AND VALUE TO DISPLAY.
*
*****************************************************************

B7000-DISPLAY-FIELD-INFO.
    ADD +1                    TO FLD-IDX.
```

```
    MOVE OUT-FLDNAME          TO WS-FIELD-NAME.
    MOVE GET-FIELD-VALUE      TO WS-FIELD-VALUE.
    DISPLAY WS-DISPLAY-FIELD.

    MOVE LOW-VALUES           TO OUT-FLDNAME
                                 GET-FIELD-VALUE.
*****************************************************************
* B8000-GET-OUTPUT-MESSAGE.
*
* GET THE POINTER TO THE OUTPUT MESSAGES.
* GET THE MESSAGE AND THE LENGTH OF THE MESSAGE.
*
* NOTE THAT THIS PROGRAM WILL ONLY HANDLE ONE OUTPUT MESSAGE.
*
* WRITE THE OUTPUT MESSAGE TO THE OUTPUT FILE.
*
*****************************************************************

B8000-GET-OUTPUT-MESSAGE.
    CALL 'GETOTMSG' USING FORMATR-PTR
                          OUT-FORMAT
                          OUT-MESSAGE-PTR.

    CALL 'MSGCOUNT' USING MSG-CNT
                          OUT-MESSAGE-PTR
                          OUT-MESSAGE-HOLD.

    CALL 'MSGOUTPT' USING OUT-MESSAGE-HOLD
                          MESSAGE-VALUE-OUT
                          MESSAGE-SIZE-OUT.

    WRITE OUTPUT-RECORD FROM MESSAGE-VALUE-OUT.

*****************************************************************
* C1000-OPEN-FILES.
*
* OPEN FILES.
*
*****************************************************************

C1000-OPEN-FILES.
```

```
    OPEN INPUT  INPUT-FILE
                FORMAT-FILE
                SESSION-FILE

         OUTPUT OUTPUT-FILE.

*****************************************************************
* C2000-READ-CONTROL-FILES.
*
* READ CONTROL FILES.
*
*****************************************************************

C2000-READ-CONTROL-FILES.

    READ FORMAT-FILE.

    UNSTRING FORMAT-RECORD
       DELIMITED BY ALL SPACE
       INTO IN-FORMAT  OF FORMAT-RECORD-FIELDS
            COUNT      IN INPUT-FORMAT-LENGTH
            OUT-FORMAT OF FORMAT-RECORD-FIELDS
            COUNT      IN OUTPUT-FORMAT-LENGTH.

    READ SESSION-FILE.

    UNSTRING SESSION-RECORD
       DELIMITED BY ALL SPACE
       INTO SESSION-NAME
       COUNT IN SESSION-LENGTH.

*****************************************************************
* C3000-READ-INPUT-FILE.
*
* READ INPUT FILE, AT END SET EOF FLAG.
*
*****************************************************************


C3000-READ-INPUT-FILE.

     READ INPUT-FILE
```

```
        AT END SET EOF TO TRUE
     END-READ.

*****************************************************************
* D1000-CLOSE-FILES.
*
* CLOSE FILES.
*
*****************************************************************

D1000-CLOSE-FILES.
    CLOSE INPUT-FILE
          OUTPUT-FILE
           FORMAT-FILE
         SESSION-FILE.
```

RULTSTCB Sample Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. RULTSTCB.
AUTHOR. NEON - NEW ERA OF NETWORKS.

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

INPUT-OUTPUT SECTION.
FILE-CONTROL.

    SELECT INPUT-FILE
        ASSIGN TO INPUTT
        FILE STATUS IS INPUT-FILE-STATUS.

    SELECT OUTPUT-FILE
        ASSIGN TO OUTFILE
        FILE STATUS IS OUTPUT-FILE-STATUS.

    SELECT APP-FILE
        ASSIGN TO APPFILE.

    SELECT SESSION-FILE
        ASSIGN TO SESSFILE.

DATA DIVISION.

FILE SECTION.
FD  INPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS V
    RECORD IS VARYING FROM 1 TO 32752 CHARACTERS
              DEPENDING ON IN-REC-LEN
    LABEL RECORDS ARE STANDARD

    DATA RECORD IS INPUT-RECORD.
```

```
01  INPUT-RECORD.
    05 INPUT-REC           PIC X(01)
                           OCCURS 1 TO 32752
                           DEPENDING ON IN-REC-LEN
                           INDEXED BY IN-IDX.

FD  APP-FILE
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F.

01  APP-RECORD            PIC X(80).

FD  SESSION-FILE
    BLOCK CONTAINS 0 RECORDS
    LABEL RECORDS ARE STANDARD
    RECORDING MODE IS F.

01  SESSION-RECORD         PIC X(80).

FD  OUTPUT-FILE
    BLOCK CONTAINS 0 RECORDS
    RECORDING MODE IS V
    RECORD IS VARYING FROM 1 TO 32752 CHARACTERS
    DEPENDING ON MESSAGE-SIZE-OUT
    LABEL RECORDS ARE STANDARD.

01  OUTPUT-RECORD.
    05 OUTPUT-REC          PIC X(01)
                           OCCURS 1 TO 32752 TIMES
                           DEPENDING ON MESSAGE-SIZE-OUT
                           INDEXED BY OUT-IDX.

WORKING-STORAGE SECTION.

01  SESSION-PTR           POINTER   VALUE NULL.
01  RULE-PTR              POINTER   VALUE NULL.
01  ALERT-PTR             PIC 9(9) VALUE 00 USAGE IS BINARY.
01  LOGFILE-PTR           PIC X(80) VALUE SPACES.
01  SESSION-NAME          PIC X(33).
01  SESSION-LENGTH        PIC 9(02).
01  APP-NAME              PIC X(33).
```

```
01  MESS-NAME              PIC X(33).
01  MESS-STRING            PIC X(33).
01  MESS-LENGTH            PIC 9(07) VALUE 32752.
01  LOAD-NOW               PIC 9(07) VALUE 0 USAGE IS BINARY.
01  RTRN-CODE              PIC 9(07) VALUE 0 USAGE IS BINARY.
01  HIT-RULE               PIC X(33).
01  NOHIT-RULE             PIC X(33).
01  HIT-RULENAME           PIC X(33).
01  NO-HIT-RULENAME        PIC X(33).
01  ACTION-CNTR            PIC 9(07) VALUE 0 USAGE IS BINARY.
01  OPTION-CNTR            PIC 9(07) VALUE 0 USAGE IS BINARY.
01  ERROR-MESS             PIC X(33).
01  ERROR-NO               PIC 9(07) VALUE 0 USAGE IS BINARY.
01  ERROR-DESC             PIC X(33).
01  A-INDX                 PIC 9(09) VALUE 1 USAGE IS BINARY.
01  O-INDX                 PIC 9(09) VALUE 1 USAGE IS BINARY.
01  S-INDX                 PIC 9(09) VALUE 1 USAGE IS BINARY.
01  OC-INDX                PIC 9(09) VALUE 1 USAGE IS BINARY.
01  OC2-INDX               PIC 9(09) VALUE 1 USAGE IS BINARY.
01  NHR-INDX               PIC 9(09) VALUE 1 USAGE IS BINARY.
01  HR-INDX                PIC 9(09) VALUE 1 USAGE IS BINARY.
01  HIT-COUNT              PIC 9(07) VALUE 1 USAGE IS BINARY.
01  NO-HIT-COUNT           PIC 9(07) VALUE 1 USAGE IS BINARY.

01  INPUT-FILE-STATUS      PIC X(02) VALUE SPACES.
01  OUTPUT-FILE-STATUS     PIC X(02) VALUE SPACES.

01  APP-NAME-LENGTH        PIC 9(02) VALUE 0.
01  MESS-NAME-LENGTH       PIC 9(02) VALUE 0.

01  APP-RECORD-FIELDS.
    05  APP-NAME           PIC X(32).
    05  MESS-NAME          PIC X(32).
    05  FILLER             PIC X(16).

01  OPTION-COUNT-TABLE.
    05  OPT-COUNT-LIST     PIC 9(09) VALUE 0 USAGE IS BINARY
                           OCCURS 1 TO 300000 TIMES
                           DEPENDING ON ACTION-CNTR.
01  HIT-RULE-TABLE.
    05  HIT-RULELIST       PIC X(33)
                           OCCURS 1 TO 20000 TIMES
```

```
                          DEPENDING ON HIT-COUNT.
01  NO-HIT-RULE-TABLE.
    05  NO-HIT-RULELIST   PIC X(33)
                          OCCURS 1 TO 20000 TIMES
                          DEPENDING ON NO-HIT-COUNT.
01  OPTION-ARRAY.
    10 TOT-OPTION         PIC X(66)
                          OCCURS 1 TO 250000 TIMES
                          DEPENDING ON OPTION-CNTR.

01  ACTION-ARRAY.
    05 ACTION-X           PIC X(33)
                          OCCURS 1 TO 100000 TIMES
                                 DEPENDING ON ACTION-CNTR.

01  SUB-ARRAY.
    05 SUB-X              PIC 9(09) VALUE 1 USAGE IS BINARY
                          OCCURS 1 TO 100000 TIMES
                          DEPENDING ON ACTION-CNTR.
01  FLAGS.
    05 EOF-FLAG           PIC X VALUE 'N'.
       88 NOT-EOF               VALUE 'N'.
       88 EOF                   VALUE 'Y'.

01  ACCUMULATORS.
    05 IN-REC-LEN         PIC 9(09) COMP VALUE ZERO.

01  MESSAGE-SIZE-OUT      PIC 9(09) VALUE 00 USAGE IS BINARY.

01  DBASE-VALUES.
    05  DB2-TYPE          PIC S9(9) VALUE 5 USAGE IS BINARY.
    05  ADABAS-TYPE       PIC S9(9) VALUE 6 USAGE IS BINARY.

01  BLANK-LINE            PIC X(80) VALUE SPACES.
01  IN-FORMAT-PTR         PIC X(33) VALUE SPACES.
01  OUT-FORMAT-PTR        PIC X(33) VALUE SPACES.

PROCEDURE DIVISION.

A0000-MAIN.
    PERFORM A1000-OPEN-FILES.
```

```
    IF INPUT-FILE-STATUS = ZEROES

       IF OUTPUT-FILE-STATUS = ZEROES

          PERFORM A2000-READ-CNTL-FILES

       IF SESSION-LENGTH > 31
           DISPLAY 'Session name:  ' SESSION-RECORD
                   ' is too long.'
           DISPLAY
             'SESSION NAME MUST BE 31 BYTES OR LESS'
       ELSE
          IF APP-NAME-LENGTH = 0
             DISPLAY 'YOU MUST GIVE AN APPLICATION GROUP NAME'
          ELSE
             IF MESS-NAME-LENGTH = 0
                DISPLAY 'YOU MUST GIVE A MESSAGE TYPE NAME'
             ELSE
                PERFORM A3000-READ-INPUT-FILE
                PERFORM A4000-START-SES.
                PERFORM A5000-CREATE-RULE-ENGINE.
                PERFORM A6000-LOAD-RULES-SET.
                PERFORM B1000-PROCESS-RULE.
                PERFORM C5000-DEL-RULE-ENGINE.
                PERFORM C6000-CLOSE-FILES.
             END-IF
          END-IF
       END-IF

    ELSE
       DISPLAY 'ERROR OPENING OUTPUT FILE. FILE STATUS:
               OUTPUT-FILE-STATUS
    END-IF

ELSE
   DISPLAY 'ERROR OPENING INPUT FILE.  FILE STATUS:
           INPUT-FILE-STATUS
END-IF
GOBACK.

A1000-OPEN-FILES.
    OPEN INPUT INPUT-FILE
```

```
               APP-FILE
               SESSION-FILE
          OUTPUT OUTPUT-FILE.

A2000-READ-CNTL-FILES.
    READ SESSION-FILE
    END-READ
    UNSTRING SESSION-RECORD
       DELIMITED BY ALL SPACE
       INTO SESSION-NAME
       COUNT IN SESSION-LENGTH.

    READ APP-FILE
    END-READ
    UNSTRING APP-RECORD
       DELIMITED BY ALL SPACE
       INTO APP-NAME  OF APP-RECORD-FIELDS
            COUNT     IN APP-NAME-LENGTH
            MESS-NAME OF APP-RECORD-FIELDS
            COUNT     IN MESS-NAME-LENGTH.

A3000-READ-INPUT-FILE.

    PERFORM UNTIL EOF

       READ INPUT-FILE
         AT END MOVE 'Y' TO EOF-FLAG
       END-READ
       MOVE INPUT-RECORD TO MESS-STRING
    END-PERFORM.

A4000-START-SES.
    CALL 'STARTSES' USING SESSION-PTR
                          SESSION-NAME
                              DB2-TYPE.
    DISPLAY 'SESSION NAME IS ' SESSION-NAME.
    DISPLAY BLANK-LINE.

A5000-CREATE-RULE-ENGINE.
    DISPLAY 'CRRULENG' USING SESSION-PTR
    CALL 'CRRULENG' USING SESSION-PTR
                          RULE-PTR.
```

```
    DISPLAY BLANK-LINE.

A6000-LOAD-RULES-SET.
    DISPLAY 'IN LRULESET'.
    CALL 'LRULESET' USING RULE-PTR
                          APP-NAME
                          MESS-NAME
                          LOAD-NOW
                          RTRN-CODE.
    DISPLAY 'APP-NAME IS ' APP-NAME.
    DISPLAY 'MESS-NAME IS ' MESS-NAME.
    DISPLAY 'LOAD-NOW IS ' LOAD-NOW.
    DISPLAY 'RETURN CODE IS ' RTRN-CODE.
    DISPLAY BLANK-LINE.

B1000-PROCESS-RULE.
    PERFORM B2000-EVAL.

    PERFORM B2500-GETRULE.

    PERFORM Z4000-GETTABLE.

B2000-EVAL.
    DISPLAY 'IN EVAL'.
    CALL 'EVAL' USING RULE-PTR
                      APP-NAME
                      MESS-NAME
                    INPUT-RECORD
                    IN-REC-LEN.
    DISPLAY 'APP NAME IS ' APP-NAME.
    DISPLAY 'MESS NAME IS ' MESS-NAME.
    DISPLAY 'MESS LENGTH IS ' IN-REC-LEN.
    DISPLAY BLANK-LINE.

B2500-GETRULE.
    DISPLAY 'IN GETRULE'.
    CALL 'GETRULE' USING RULE-PTR
                         HIT-COUNT
                         NO-HIT-COUNT
                         HIT-RULE-TABLE
                         NO-HIT-RULE-TABLE.
    DISPLAY 'NUMBER OF NO HIT RULES IS ' NO-HIT-COUNT.
```

```
    DISPLAY 'NUMBER OF HIT RULES IS ' HIT-COUNT.
    DISPLAY BLANK-LINE.
    MOVE 1 TO NHR-INDX.
    MOVE 1 TO HR-INDX.
    DISPLAY '------NO HIT RULES------'.
    PERFORM UNTIL NHR-INDX > NO-HIT-COUNT
       DISPLAY NO-HIT-RULELIST(NHR-INDX)
       ADD 1 TO NHR-INDX
    END-PERFORM.
    DISPLAY '-------HIT RULES-------'.
    PERFORM UNTIL HR-INDX > HIT-COUNT
       DISPLAY HIT-RULELIST(HR-INDX)
       ADD 1 TO HR-INDX
    END-PERFORM.
    DISPLAY BLANK-LINE.


C5000-DEL-RULE-ENGINE.
    DISPLAY 'IN DLRULENG'.
    CALL 'DLRULENG' USING RULE-PTR.
    DISPLAY BLANK-LINE.

C6000-CLOSE-FILES.
    CLOSE INPUT-FILE
          APP-FILE
          SESSION-FILE
          OUTPUT-FILE.

Z4000-GETTABLE.
    MOVE ZEROS TO OPTION-COUNT-TABLE.
    MOVE SPACES TO ACTION-ARRAY.
    MOVE SPACES TO OPTION-ARRAY.
    DISPLAY 'IN Z4000-GETTABLE'.
    CALL 'LOADTAB' USING RULE-PTR
                         ACTION-CNTR
                         OPTION-CNTR
                         OPTION-COUNT-TABLE
                         ACTION-ARRAY
                         SUB-ARRAY
                         OPTION-ARRAY.
    DISPLAY 'THERE ARE ' ACTION-CNTR ' ACTIONS'.
    DISPLAY 'THERE ARE ' OPTION-CNTR ' OPTIONS'.
```

```
    DISPLAY BLANK-LINE.
    MOVE 1 TO A-INDX.
    MOVE 1 TO S-INDX.
    MOVE 1 TO OC-INDX.
    DISPLAY 'SUBSCRIPTION       ACTION
-           'OPTION'.
    PERFORM UNTIL A-INDX> ACTION-CNTR
      DISPLAY  SUB-X(S-INDX)
      DISPLAY '                  ' ACTION-X(A-INDX)
      MOVE 1 TO OC2-INDX
      PERFORM UNTIL OC2-INDX > OPT-COUNT-LIST(OC-INDX)
        DISPLAY '                                    '
-           '  ' TOT-OPTION(O-INDX)
        ADD 1 TO O-INDX
        ADD 1 TO OC2-INDX
      END-PERFORM
    DISPLAY BLANK-LINE
    DISPLAY'************************************************'
-   '************************'
    DISPLAY BLANK-LINE
    ADD 1 TO A-INDX
    ADD 1 TO S-INDX
    ADD 1 TO OC-INDX
    END-PERFORM.
```

TESTCOB1 Sample Program

```
IDENTIFICATION DIVISION.
PROGRAM-ID. TESTCOB1.
AUTHOR. NEON - 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.

A0000-CALL-USER-EXIT-APIS.

    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 B1000-MOVE-DATA-TO-OUTPUT.

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 B1000-MOVE-DATA-TO-OUTPUT.

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 B1000-MOVE-DATA-TO-OUTPUT.

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 B1000-MOVE-DATA-TO-OUTPUT.

    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 B1000-MOVE-DATA-TO-OUTPUT.

    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 B1000-MOVE-DATA-TO-OUTPUT.

    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.

    GOBACK.

B1000-MOVE-DATA-TO-OUTPUT.

    IF WS-DATA-LENGTH > WS-MAX-DATA-LENGTH
        MOVE WS-MAX-DATA-LENGTH   TO WS-DATA-LENGTH
    END-IF.

    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.
```

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 MQSeries Integrator books, are trademarks of International Business Machines Corporation in the United States, or other countries, or both:

MQSeries
OS/390
AIX
DB2
IBM

NEONFormatter and NEONRules 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 X/Open Company Limited.

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

Index

G

L

M

N

O

P

R

S

T

U

V

Sending your comments to IBM
MQSeries Integrator for OS/390
Program Supplement
GC34-5875-00

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
MQSeries Integrator for OS/390
Program Supplement
GC34-5875-00

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

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR
NO STAMP REQUIRED

IBM

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____
Company or Organization _____
Address _____
_____
EMAIL _____
Telephone _____

4 Fasten here with adhesive tape

Cut along this line

Printed in U.S.A

GC34-5875-00