WebSphere MQ Integrator for z/OS - Writing New Era of Networks COBOL user exits

Version 1.0

14 December, 2001

Roy Saxton
IBM United Kingdom Limited
Hursley Park, Winchester
Hampshire S021 2JN

rsaxton@uk.ibm.com

Take Note!

Before using this report be sure to read the general information under "Notices".

First Edition, December 2001

This edition applies to Version 1.0 of *WebSphere MQ Integrator for z/OS - Writing New Era of Networks COBOL user exits* and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Any reference to an IBM licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product.

Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS-IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Trademarks and service marks

The following terms, used in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- IBM
- OS/390
- WebSphere MQ Integrator
- z/OS

The following terms are trademarks of other companies:

Adobe Acrobat	Adobe Systems Incorporated
New Era of Networks	New Era of Networks Incorporated

Acknowledgments

The following people contributed significantly to the development of the sample programs that are included with this SupportPac.

James J. Di Vincenzo, Sybase Inc.

Jim Roohan, Sybase Inc.

Summary of Amendments

Date	Changes
14 December 2001	Initial release

Preface

MQSeries Integrator for OS/390 Version 1.1 provided the capability of adding user-written C++ and COBOL programs as formatter exit routines to customize the way it reformatted messages. If you have existing user exits that you developed for MQSeries Integrator for OS/390 Version 1.1, they will continue to work with WebSphere MQ Integrator for z/OS Version 2.1 when you recompile them for the new environment, provided that their designs meet certain programming constraints. These include Language Environment restrictions for COBOL user exits running under UNIX System Services, and database access limitations for both COBOL and C++ user exits. In addition, special techniques must be used for dynamic allocation and unallocation of datasets. If your existing exits do not follow these constraints, you may have to modify them before they will work with WebSphere MQ Integrator for z/OS Version 2.1.

If you are creating new user exits specifically for WebSphere MQ Integrator for z/OS Version 2.1, you must design the exits with these constraints in mind.

SupportPac ID11 outlines the constraints for user exits written in COBOL in particular and provides programming examples to show how user exits should be written to work with WebSphere MQ Integrator for z/OS Version 2.1.

Bibliography

- *OS/390 MVS Programming: Authorized Assembler ServIces Guide,* IBM Corporation. GC28-1763.

- *Cobol for OS/390 and VM Programming Guide Version 2 Release 2,* IBM Corporation. SC26-9049.

- *DB2 Universal Database for OS/390 and z/OS Application Programming and SQL Guide Version 7*, IBM Corporation. SC26-9933.

Chapter 1. Installing the SupportPac

Contents of the SupportPac

When you unzip the downloaded id11.zip file you will find the following parts:

• id11.source.xmit	Partitioned dataset in XMIT format containing sample programs
• id11.jcl.xmit	Partitioned dataset in XMIT format containing sample JCL
• id11.dbrmlib.xmit	Partitioned dataset in XMIT format containing a dummy member
• id11.pdf	This User Guide in pdf format
• wmqi56_userexit.pdf	New Era of Networks COBOL User Exit Supplement in pdf format
• license2.txt	Licensing information for this SupportPac

The documentation

The documentation is supplied in Adobe Acrobat (.pdf) format. Move the documentation to a convenient location on your workstation from where you can launch your Acrobat Reader.

Restoring the libraries

To restore the libraries containing the sample programs and JCL, do the following:

- Allocate a dataset on the target OS/390 or z/OS machine with the following characteristics:

```
Organization  . . . : PS
Record format . . . : FB
Record length . . . : 80
Block size  . . . . : 3120
1st extent blocks . : 45
Secondary blocks  . : 11
```

- Transfer file `id11.source.xmit` to the dataset you have just created by ftp in binary format
- Issue the following TSO command on the target OS/390 or z/OS system:

```
receive indsn('myindsn') -
```

where `myindsn` is the name of the dataset you created in the first step.

- Reply to the prompt for restore parameters with `da('<HLQ>.SOURCE'),` where `<HLQ>` is the high level qualifier you want to specify for the ID11 sample source programs. You do not need to create the library before issuing the receive command.
- Repeat the procedure for file `id11.jcl.xmit`. You can reuse the file you created for ftp transfer of the first xmit format file. Enter `da('<HLQ>.JCL')` when you reply to the restore parameters, where `<HLQ>` is the high level qualifier you want to specify for the ID11 sample JCL members.

- Repeat the procedure for file `id11.dbrmlib.xmit`. You can reuse the file you created for ftp transfer of the first xmit format file. Enter `da('<HLQ>.DBRMLIB')` when you reply to the restore parameters, where `<HLQ>` is the high level qualifier you want to specify for the ID11 DBRMLIB.

Contents of the libraries

When you have received the libraries containing the sample programs and the JCL, you will find they contain the following members.

Sample source programs in <HLQ>.SOURCE:

ID11ASP1	Assembler program that dynamically allocates and unallocates a VSAM dataset
ID11CBP1	COBOL program that calls assembler routine ID11ASP1
ID11CBP2	COBOL program that makes SQL calls to read a database

Sample JCL members in <HLQ>.JCL:

ID11ASJ1	JCL for compilation of ID11ASP1
ID11CBJ1	JCL for compilation of ID11CBP1
ID11CBJ2	JCL for compilation of ID11CBP2
ID11CBJP	JCL Procedure for precompiling ID11CBP2

Dummy member in <HLQ>.DBRMLIB:

DUMMY	Dummy member to allow the DBRMLIB library to be packaged in XMIT format.

Customizing the JCL and source programs

You must customize the JCL and source programs for your installation before submitting the jobs to compile any of the programs. Instructions for making the substitutions are given in the comments in each member.

Compiling the sample source programs

ID11ASP1

Submit JCL member ID11ASJ1 to compile and link edit this sample assembler program into an SMS managed location, for example a standard load library. The load library should have the following characteristics:

```
Organization  . . . : PO

Record format . . . . U

Record length . . . : 0

Block size  . . . . : 32760
```

You will statically link edit this assembler routine with COBOL program ID11CBP1 when you submit JCL member ID11CBJ1.

ID11CBP1

Submit JCL member ID11CBJ1 to compile program ID11CBP1 and link edit it with assembler routine ID11ASP1. The executable is created in an HFS location under UNIX System Services, from where it can be invoked as a COBOL user exit. Make sure that the job returns condition code 0. Then go to the HFS directory where you have created the executable, and run the shell command `ls -l` to show the permissions on the file. The execute permission bits must be set to allow the started task ID under which the broker will run to execute the program.

In order to use ID11CBP1, you must create a sequential file that contains 80 byte data records to be sent to stdout when the exit runs. A single record saying “HELLO WORLD” will suffice.

ID11CBP2

Before compiling program ID11CBP2, you must first create a copybook member `NNFLITRL` using DCLGEN. `NNF_LITRL` is a table in the New Era of Networks formats database that can be used to test DB2 database access from within the COBOL program. The following figure shows an example DCLGEN panel for generating member `NNFLITRL`:

```
                         DCLGEN                             SSID: DEL0

===>


Enter table name for which declarations are required:

1  SOURCE TABLE NAME ===> NNF_LITRL                       (Unqualified)

 2  TABLE OWNER ..... ===> MI00USR                            (Optional)

 3  AT LOCATION ..... ===> DSN610PL                           (Optional)


Enter destination data set:          (Can be sequential or partitioned)

 4  DATA SET NAME ... ===> 'ARGO.ID11.SOURCE(NNFLITRL)'

 5  DATA SET PASSWORD ===>           (If password protected)


Enter options as desired:

 6  ACTION .......... ===> REPLACE   (ADD new or REPLACE old declaration)

 7  COLUMN LABEL .... ===> NO        (Enter YES for column label)

 8  STRUCTURE NAME .. ===>                                    (Optional)

 9  FIELD NAME PREFIX ===>                                    (Optional)

10  DELIMIT DBCS .... ===> YES       (Enter YES to delimit DBCS identifiers)

11  COLUMN SUFFIX ... ===> NO        (Enter YES to append column name)

12  INDICATOR VARS .. ===> NO        (Enter YES for indicator variables)
```

In this example, `MI00USR` is the SQL ID under which the New Era of Networks database was created, and `DSN610PL` is the DB2 location of the database. Set the destination dataset, shown in this example as `'ARGO.ID11.SOURCE(NNFLITRL)'`, to member `NNFLITRL` in the partitioned dataset where you installed SupportPac ID11. This is to allow the member to be included as a copybook when you compile program ID11CBP2.

When you have prepared copybook `NNFLITRL`, submit JCL member ID11CBJ2 to precompile, compile and linkedit program ID11CBP2 into an HFS location under UNIX System Services. Make sure that the job returns condition code 0. Then go to the HFS directory where you have created the executable, and run the shell command `ls -l` to show the permissions on the file. The execute permission bits must be set to allow the started task ID under which the broker will run to execute the program.

Finally, you will need to bind your COBOL user exit into a DB2 plan before you can use it to access the database.

Defining New Era of Networks formats

In order to use a COBOL user exit, you must define an output format with a field whose output control includes the user exit in its output operations. COBOL user exits require the name of the exit to be equal to the name of the executable. The executables that you can prepare with the SupportPac ID11 sample programs are named ID11CBP1 and ID11CBP2, so you must also define your user exits to have these names.

Chapter 2. Programming constraints for COBOL user exits

A number of constraints exist for New Era of Networks COBOL user exits that execute under UNIX System Services. These purpose of this Chapter is to outline these constraints. The sample programs included with SupportPac ID11 may be used as models for writing further exits that comply with these requirements.

Language Environment restrictions for COBOL user exits

COBOL user exits must be coded to comply with the Language Environment restrictions that exist for COBOL programs running under OS/390 Unix. Full details are given in "COBOL for VM and OS/390 Version 2 Release 2 Programming Guide", Document Number SC26-9049-05.

The most important restriction is that you cannot run a COBOL program compiled under COBOL for VM and OS/390 Version 2 Release 2 in more than one thread under UNIX System Services. Such a program can only run under exactly one process thread, and that thread must be the same for all invocations. If a process attempts to start another COBOL program in a second thread, a software condition will be raised by the COBOL runtime.

What this means in practice is that any message flow that causes a COBOL user exit to be invoked must be assigned to its own, dedicated Execution Group. If the Execution Group contains more than one message flow that invokes a COBOL user exit, it will fail with a Language Environment runtime error message when the second message flow invokes the exit. Also, you must not specify additional instances for such message flows through their properties panels, since each additional instance will cause a new thread to be created.

Database access

In general, you are recommended to use Database nodes if you wish to access databases in WebSphere MQ applications. If database access is essential for your COBOL user exits, you must observe the restrictions that are outlined here.

Database attachment model

Applications in a single address space cannot use both CAF and RRSAF to connect to DB2. Since WMQI and New Era Support connect to DB2 using RRSAF, it means that you must also use RRSAF to connect to DB2 in your user exits. If you currently use CAF, you will need to alter your source code to meet the requirements of RRSAF when you migrate to WMQI V2.1 for z/OS. You will then need to recompile and link the application specifying ATTACH(RRSAF) when you run the DB2 precompiler program, as illustrated in sample ID11CBJP. Further details are given in "DB2 Universal Database for OS/390 and z/OS Application Programming and SQL Guide Version 7", Document Number SC26-9933-00.

Database connections

DB2 does not allow multiple connections in a single processing thread. Since user exits run in the same thread as the message flow from which they are called, they can only connect to the same DB2 instance as the broker and New Era nodes.

The dsnaoini file

The Broker's `dsnaoini` file defines the characteristics of database connections that will be made in the Broker environment. There can be only one `dsnaoini` file for any Broker environment. It follows that the Broker and New Era nodes must use the same `dsnaoini` file, addressed by the `DSNAOINI` environment variable. The COBOL user exit must be bound into the plan that is specified in `dsnaoini` by the keyword `PLANNAME`. Note that the `dsnaoini` file is created from information in the `mqsicompcif` file during the broker customization step, and should not be edited manually.

You can restrict access to the COBOL user exit by binding it into a plan that also includes the DSNAOCLI packages that are required by the Broker and specifying a unique name for the plan. The following extract from a BIND job shows how to bind a COBOL user exit into a package, and the package into a plan:

```
 BIND PACKAGE(TESTEXIT)               -
      MEMBER(ID11CBP2)                -
      ACTION(REPLACE)                 -
      ISOLATION(CS)                   -
      DYNAMICRULES(RUN)               -
      ENABLE(RRSAF)                   -
      LIBRARY('ARGO.ID11.DBRMLIB')


BIND PLAN(ID11CBP2)                   -
 PKLIST(DSNAOCLI.DSNCLICS             -
        DSNAOCLI.DSNCLINC             -
        DSNAOCLI.DSNCLIRR             -
        DSNAOCLI.DSNCLIRS             -
        DSNAOCLI.DSNCLIUR             -
        DSNAOCLI.DSNCLIC1             -
        DSNAOCLI.DSNCLIC2             -
        DSNAOCLI.DSNCLIF4             -
        DSNAOCLI.DSNCLIMS             -
        DSNAOCLI.DSNCLIQR             -
        TESTEXIT.ID11CBP2        )    -
     OWNER(ARGOUSR)                   -
     QUALIFIER(MI00USR)               -
     RETAIN                           -
     ISOLATION(CS) ACTION(REPLACE)
```

You would enter the following stanza in the `mqsicompcif` file to specify ID11CBP2 as the plan in `dsnaoini`:

```
DB2_DSNACLI_PLANNAME='ID11CBP2'
```

You can restrict access to the plan by issuing the following SQL statement using SPUFI or a batch job:

```
GRANT EXECUTE ON PLAN ID11CBP2 TO <USER> ;
```

If you have multiple different COBOL exits and it is necessary to restrict access to different authorized users, it follows that you must use a different Broker environment for each set of authorized users. The `dsnaoini` file for each broker will specify a plan for which a single set of users will have execute authority.

Database updates

COBOL and C++ user exits can safely use readonly access to DB2 databases. If they make updates to a database, however, they must not issue any COMMIT or ROLLBACK calls. This is because they execute in the same unit of work as the message flow from which they have been invoked, and these calls will therefore affect all updates to recoverable resources that the message flow itself has made. Any updates made by a user exit will automatically be committed or rolled back when the message flow completes its processing.

Dynamic allocation and unallocation of datasets

COBOL applications running in the native z/OS environment typically allocate datasets using jobstep statements. This method is not available to COBOL User Exits running under UNIX System Services, and they must allocate and deallocate datasets dynamically at runtime. A convenient way to do this is to call an assembler routine that issues a DYNALLOC macro to perform this function. A sample assembler routine, ID11ASP1, and a COBOL program that calls it, ID11CBP1, are included with this SupportPac.

Exception handling

It is essential that you develop routines to handle every exception that could occur in your user exit before you put it into production. ***Unhandled exceptions in a user exit can cause the Broker to fail.*** In such a case, it may be necessary to stop and restart the Broker to recover from the failure .

Chapter 3. Problem determination

When you create a new COBOL user exit, you may typically get 0C1 and 4039 abends and coredumps written to the home directory of the started task ID in the early stages of development. Look first in the SDFS SYSLOG to see if there is an obvious cause for the abend. For example, the execute permission bits may not have been set correctly on the user exit executable. You would then see something like the following:

```
ICH408I USER(MI00USR ) GROUP(TSOUSER ) NAME(ARGOUSER, F (FRED)      )

  /u/rsaxton/D21NOV01/bin/COBTEST1

  CL(FSOBJ   ) FID(01D7D3E2E3F0F5002D07000007330000)

  INSUFFICIENT AUTHORITY TO OPEN

  ACCESS INTENT(--X)  ACCESS ALLOWED(GROUP R--)
```

If the SYSLOG does not give a specific reason for the abend, you should use IPCS to examine the traceback in the dump. This can often indicate the point of failure. For example:

- If the traceback shows an entry for `NNFOutCtrlUserExit::Apply(NNFControlData*)` before the exception, it is likely that you have not included the path to the user exit executable in the `LIBPATH` that you set in `mqsicompcif`.
- If the traceback shows an entry for `dllload` just before the exception, this indicates that the COBOL user exit could not be loaded as a DLL. Check that you specified DLL in the COBOL compilation PARMs.
- If the traceback shows no entry corresponding to the name of your user exit, the exit did not execute at all.
- If the traceback shows an entry for your user exit followed by entries for IGZ* functions, it is likely that a COBOL API call failed.

When you have established that the user exit has been successfully called, you can carry out further diagnosis by including DISPLAY statements in the program. This should cause output to be sent to file `stdout` in the Broker output directory.

Chapter 4. COBOL user exit checklist

This Chapter gives some points for your attention when you are preparing COBOL user exits to work with WebSphere MQ Integrator for z/OS Version 2.1.

Creating the COBOL user exit

- If your exit uses DB2 access:

    1. Run DCLGEN to create any necessary COBOL copybooks before precompiling the program to translate the EXEC SQL statements.

    2. Don't forget to code `MOVE +0 TO RETURN-CODE-VALUE` before exiting your program if processing was successful, or the message will be sent to the failure queue.

    3. Bind the compiled program into a plan and specify the name of the plan in `mqsicompcif`. The plan must also contain the DSNACLI packages needed for the DB2 CLI. You must first stop the broker if it is running, because it will have a lock on the plan.

- Compile and link your COBOL user exit routine into an HFS directory under USS. Check the permission bits to make sure it is executable by the started task ID.

Developing the user exit

During iterative development of your user exit:

- Stop and start your broker to bring in the new version of the user exit every time you recompile it.

- Look out for coredumps accumulating in the home directory of the started task ID. These will typically be produced by 0C1 or 4039 abends. Each may be several hundred megabytes in size.

- Develop routines to handle every exception that might occur within the user exit. ***Unhandled exceptions in a user exit can cause the Broker to fail.***

Defining formats

- Define your COBOL User Exit Output Operation to have the same name as the executable.

- Make sure that the COBOL user exit does not have the same name as a C++ user exit function.

Configuring the environment

- Editing file `mqsicompcif`:

    1. `LIBPATH` must include the path to your user exit executable.

    2. If you access DB2 from within the exit, `DB2_DSNACLI_PLANNAME` must specify the plan in which your COBOL user exit is bound.

    3. `DB2_TABLE_OWNER` must specify the SQL ID under which the Broker database was created, and **not** the SQL ID under which the New Era of Networks database was created.

    4. Set the environment variables `NNSY_ROOT`, `NNSY_CONFIG_FILE_PATH` and `NNSY_CATALOGUES` to appropriate values for your installation.

- Editing file `nnsyreg.dat`:

    1. Identifier `Session.MQSI_PLUGIN` must specify the SQL ID under which the New Era of Networks database was created.