Micro Focus Enterprise Developer チュートリアル

メインフレーム COBOL 開発： MQ メッセージ連携

1. 目的

本チュートリアルでは、CICS から入力したメッセージを MQ へ連携する方法の習得を目的としています。

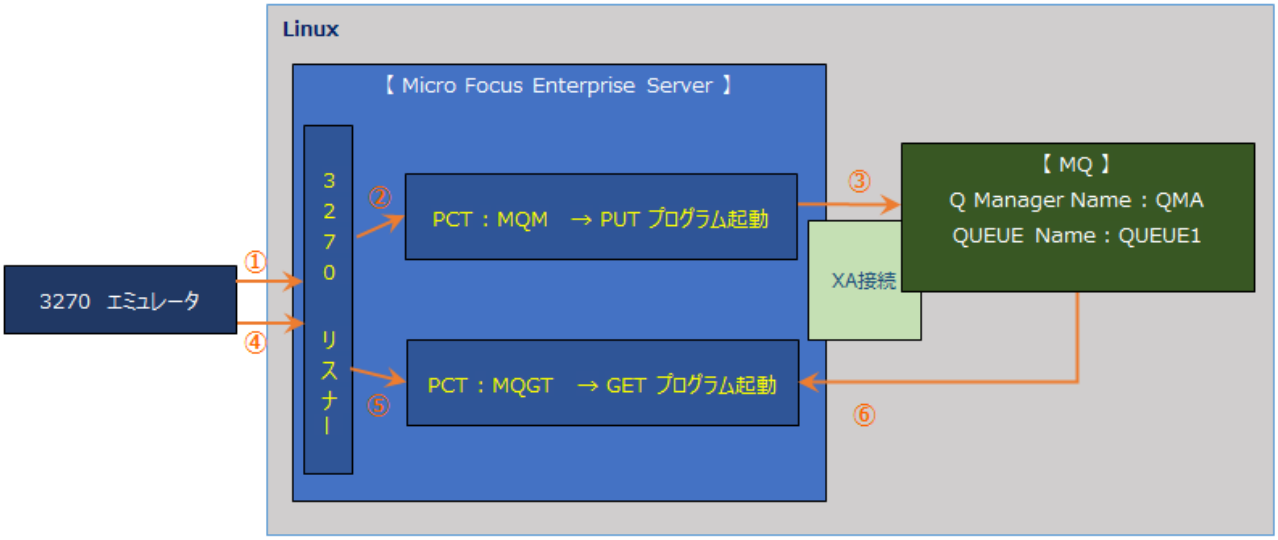
2. 前提

- 使用した OS ： Red Hat Enterprise Linux Server release 7.2 x64
- 使用した WebSphere MQ ： IBM WebSphere MQ 7.5.0.1 64-Bit
- 使用した TN3270 エミュレータ ： Micro Focus Rumba 8.1.0
- 使用マシンに Micro Focus Enterprise Developer 4.0 for Linux and UNIX がインストールされていること。

- WebSphere MQ に対象とするキュー・マネージャーとキューが設定されており、正常に開始されていること。
- CICS チュートリアルを終了していること。
- Linux/UNIX チュートリアルを終了していること。

3. 実施概要

下記図の手順で MQ メッセージ連携を行います。



4. 環境変数の設定

実行にあたり、次のように環境変数を設定する必要があります。

1）SJIS ロケールの指定

コマンド例）export LANG=ja_JP.sjis

2）MQ 環境の指定

コマンド例）. /opt/mqm/bin/setmqenv -s

3）COBOL 実行環境の指定

コマンド例）. /opt/mf/ED40PU2/bin/cobsetenv

4）COBOL 動作モードの指定

コマンド例）export COBMODE=64

本チュートリアルでは 64 ビットを使用しています。32 ビット指定も可能ですが、関連製品や実行ファイルのビット数は一致させる必要があります。

5）MQ の COPY ファイルパスを指定します。

コマンド例）export COBCPY=/opt/mqm/inc/cobcpy64:$COBCPY

6）MQ の ロードライブラリパスを指定します。

コマンド例）export LD_LIBRARY_PATH=/opt/mqm/lib64:$LD_LIBRARY_PATH

5. Enterprise Server インスタンスと MQ の連携方法

MQ 連携は XA リソースを使用した方法を推奨しています。MQ のトリガー機能をご利用の場合は MQ リスナーが必要になります。CICS で使用する SIT に MQ を指定して関連付けることも可能ですが、本チュートリアルでは XA 接続を実施していきます。連携方法の詳細は下記 URL をご参照ください。

https://www.microfocus.co.jp/manuals/ED40/Eclipse/GUID-C71C8519-E360-4F51-9701-69AF5A086022.html

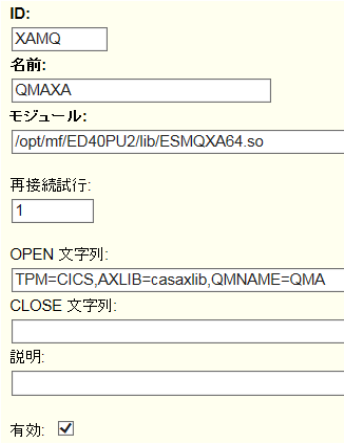
CICS インスタンスの構築に関しては［CICS チュートリアル］をご参照ください。

1）製品に含まれている MQ 用の XA リソースを Enterprise Server インスタンスへ指定します。

管理画面から対象インスタンスが停止状態であることを確認して、[編集]ボタン ＞ [XA リソース] タブ ＞ [追加] ボタンをクリックします。

２）下記の内容を登録して［追加］ボタンをクリックします

項目名	説明
ID	4 桁の ID を指定します。ここでは XAMQ を指定します。
名前	任意の名前を指定します。ここでは QMAXA を指定します。
モジュール	XA リソースのフルパスを指定します。$COBDIR/lib に存在する 64 ビット用の ESMQXA64.so（32 ビット用は ESMQXA.so）を使用します。 例）/opt/mf/ED40PU2/lib/ESMQXA64.so
OPEN 文字列	キュー・マネージャーなど、必要な情報を指定します。 例）TPM=CICS,AXLIB=casaxlib,QMNAME=QMA
有効	チェックをオンにします。



詳細は下記 URL をご参照ください。

https://www.microfocus.co.jp/manuals/ED40/Eclipse/GUID-7BA68541-518B-46C3-A751-4BCB03BA77F4.html

３）対象インスタンス開始後コンソールログを表示し、下記のように MQ インターフェイスが正常にロードされていることを確認してください。インスタンス開始前に MQ が開始されていなければなりません。

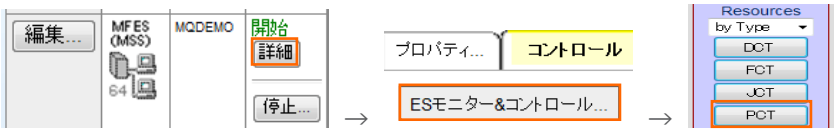
```
CASXC0020I XAMQ XA interface loaded. Name(MQSeries_XA_RMI), Registration Mode(Dynamic) 14:03:44
CASXC0015I XAMQ XA interface initialized successfully 14:03:44
```

注意）管理画面の開始ボタンを使用する際は casperm コマンドで設定するユーザがプロセスオーナーとなり、 casstart コマンドを使用する際はコマンド実行ユーザがプロセスオーナーとなるため、このユーザが MQ の管理権限を持つグループに属する必要があります。たとえば mqm をグループとすれば、開始ユーザアカウントはこのグループに所属していなければなりません。

6. PUT プログラムの準備と実行

１） メッセージの PUT に使用する PCT と、これに呼ばれる CICS プログラムを準備します。

① 管理画面から対象インスタンスの［詳細］ボタン > ［ES モニター＆コントロール］ボタン > Resources の［by Type］を選択後、［PCT］ボタンをクリックします。



② CICS の SIT へ指定したグループへ PCT を追加して［Program Name］へ CICS プログラム名を指定します。ここではサンプルプログラムの MQ00 を指定します。 MQ01 も同様に作成し、最後に［Install］ボタンをクリックしてください。

③ PCT から呼ばれる MQ00 プログラムではマップの SEND を行い MQ01 プログラムを呼び出しています。

```
EXEC CICS SEND ↵
    MAP('MQMNU')  ↵
    CURSOR(590)↵
    MAPSET('MQSET') FREEKB ↵
    ERASE MAPONLY     ↵
END-EXEC.↵
トランザクションの呼び出し    ↵
EXEC CICS RETURN TRANSID('MQ01') END-EXEC.
```

④ MQ01 プログラムでは入力値を判定後、MQ へ PUT する MQ02 プログラムを呼び出しています。

```
MQ への処理↵
EXEC CICS LINK PROGRAM('MQ02')
    COMMAREA(PGVALI) ↵
    LENGTH(20) ↵
END-EXEC. ↵
```

⑤ MQ02 プログラムへは MQ にメッセージを PUT するため、MQ に含まれる下記のコピー文を WORKING-STORAGE SECTION へ指定します。

COPY CMQV.

COPY CMQODV.

COPY CMQMDV.

COPY CMQPMOV.

PROCEDURE DIVISION では作成したキュー名を指定して MQ をオープンします。

```
 MOVE 'QUEUE1' TO MQOD-OBJECTNAME.
 ADD MQOO-OUTPUT MQOO-FAIL-IF-QUIESCING
          GIVING OPTIONS.
 CALL 'MQOPEN'
   USING HCONN, OBJECT-DESCRIPTOR,
   OPTIONS, Q-HANDLE,
   OPEN-CODE, REASON.
```

このプログラムでは CICS から入力されたメッセージを MQVALUE に保持しているため、この値を BUFFER へ転送して MQ へ PUT します。

```
 MOVE MQPMO-NO-SYNCPOINT TO MQPMO-OPTIONS.
 MOVE 60 to BUFFER-LENGTH.
 MOVE MQVALUE TO BUFFER.
 CALL 'MQPUT'
   USING HCONN, Q-HANDLE,
   MESSAGE-DESCRIPTOR, PMOPTIONS,
   BUFFER-LENGTH, BUFFER,
   COMPLETION-CODE, REASON.
```

MQ をクローズします。

```
MOVE MQCO-NONE TO OPTIONS.
CALL 'MQCLOSE'
  USING HCONN, Q-HANDLE, OPTIONS,
  COMPLETION-CODE, REASON.
```

PUT 内容を確認するため、CICS WRITE OPERATOR を実行してコンソールログへ内容を出力します。

```
EXEC CICS WRITE OPERATOR TEXT(WS-TEXT)
          TEXTLENGTH(WS-TEXT-LEN) END-EXEC
```

⑥ 前述の 3 プログラムをコンパイルして生成された実行ファイルを、対象インスタンスに指定した CICS トランザクションパスへ配置します。BMS は Linux ではコンパイルできませんので、Windows 開発環境で生成した MOD ファイルを転送しておきます。

コンパイルコマンド例）cob -u MQ00.cbl -C "DIALECT(MF) OSVS CHARSET(ASCII) CICSECM() COPYEXT(,cpy)"

2）PCT へ登録したトランザクションを起動して画面からメッセージを入力します。

下記の例では TEST MSG をメッセージとして PUT します。終了は通信を切断してください。

3）CICS WRITE OPERATOR によって出力された内容をコンソールログで確認します。

```
MQDEMO   CASOP0000I  From (SYSAD,B000,MQ01) MQ PUT VALUE :      TEST MSG
```

4）キューの内容を確認します。

MQ エクスプローラーから内容を確認します。画面で入力した値が格納されています。

5） CICS 画面から入力した値が MQ の指定したキューへ正常に PUT されたことが確認できました。

7. GET プログラムの準備と実行

1）GET プログラムを呼び出す PCT と、この GET プログラムを準備します。

① PUT と同様に CICS の SIT へ指定したグループへ PCT を追加して [Program Name] へ CICS プログラム名を指定します。ここではサンプルの MQGETWRT を指定します。

② PCT から呼ばれる [MQGETWRT] プログラムでは、MQ メッセージを GET する MQMSGGET プログラムを呼び出しています。

```
EXEC CICS LINK PROGRAM('MQMSGGET')
END-EXEC.
```

③ MQMSGGET プログラムでは、MQ に含まれる下記のコピー文を WORKING-STORAGE SECTION へ指定します。

```
COPY CMQV.
COPY CMQODV.
COPY CMQMDV.
COPY CMQGMOV.
```

PROCEDURE DIVISION では作成したキュー名を指定して MQ をオープンします。

```
MOVE 'QUEUE1' TO MQOD-OBJECTNAME.
ADD MQOO-INPUT-AS-Q-DEF MQOO-FAIL-IF-QUIESCING
          GIVING OPTIONS.
CALL 'MQOPEN'
  USING HCONN, OBJECT-DESCRIPTOR,
  OPTIONS, Q-HANDLE,
  OPEN-CODE, REASON.
```

GET する間隔や長さを指定後、キューからメッセージを GET します。

```
    MOVE MQMI-NONE TO MQMD-MSGID.
    MOVE MQCI-NONE TO MQMD-CORRELID.
    MOVE SPACES TO BUFFER.
    ADD MQGMO-WAIT MQGMO-NO-SYNCPOINT GIVING MQGMO-OPTIONS.
** ミリ秒
    MOVE 10000 TO MQGMO-WAITINTERVAL.
    MOVE 64 to BUFFER-LENGTH.

    CALL 'MQGET'
     USING HCONN, Q-HANDLE,
     MESSAGE-DESCRIPTOR, GMOPTIONS,
     BUFFER-LENGTH, BUFFER, DATA-LENGTH,
     COMPLETION-CODE, REASON.
```

GET したメッセージがスペース以外の場合は、このメッセージを利用するプログラムを呼び出すロジックが下記になります。

```
IF BUFFER IS NOT EQUAL TO SPACE
   MOVE BUFFER TO MQVALUE
メッセージを利用するプログラムの呼び出し箇所
     EXEC CICS LINK PROGRAM('XXXXX')
          COMMAREA(MQVALUE)
          LENGTH(20)
     END-EXEC
END-IF.
```

MQ をクローズします。

```
MOVE MQCO-NONE TO OPTIONS.
CALL 'MQCLOSE'
 USING HCONN, Q-HANDLE, OPTIONS,
 COMPLETION-CODE, REASON.
```

④ 前述の 2 プログラムをコンパイルして生成された実行ファイルを対象インスタンスに指定した CICS トランザクションパスへ配置します。

コンパイルコマンド例）

cob -u MQMSGGET.cbl -C"DIALECT(ENTCOBOL) CICSECM() CHARSET(ASCII) COPYEXT(,cpy)"

2）3270 エミュレータから PCT で登録したトランザクションを起動します。

これにより GET プログラムが起動されます。通信の切断により終了させます。

3）MQMSGGET プログラムの中で指定している CICS WRITE OPERATOR によって出力された内容をコンソールログで確認します。

MQ から GET したメッセージが PUT メッセージと同様であることが確認できます。

```
MQDEMO   CASOP0000I  From (SYSAD,A000,MQ01) MQ PUT VALUE :      TEST MSG          09:37:56
MQDEMO   CASOP0000I  From (SYSAD,B000,MQGT) MQ message is :     TEST MSG          10:15:52
MQDEMO   CASOP0000I  From (SYSAD,B000,MQGT) MQ no more messages                    10:16:02
```

4）キューの内容を確認します。

再度、MQ エクスプローラーから内容を確認します。 PUT メッセージが GET により消去されました。

8. まとめ

Enterprise Server インスタンスに登録した PCT プログラムを利用して 3270 エミュレータから入力したメッセージが MQ のキューへ書かれ、このメッセージを同じく PCT プログラムから取得する方法を確認できました。

WHAT'S NEXT

- メインフレーム COBOL 開発 ： CICS SIT 構築