QCOPYPRP

Sample COBOL CICS WMQ Program
Copy messages and apply a property

The IBM ATS WebSphere MQ team:

Lyn Elkins – elkinsc@us.ibm.com
Mitch Johnson – mitchj@us.ibm.com

Introduction

This document describes the sample COBOL MQ CICS program, QCOPYPRP. This program copies messages from a source queue to a target queue and applies message a message property to each message that is copied. This sample requires WMQ V7.0.1 or above and CICS 3.2 or above.

Please note that the following PTFs, or their equivalent for your release level, need to be applied to support the WMQ V7 verbs, and should be applied before implementing these samples

CICS TS 3.2 – PK66866 (UK52671,UK52672,UK52673,UK52680) OR
CICS TS 4.1 – PK89844 (UK52619,UK52667,UK52668,UK52669)

The program is single purpose, it was created to demonstrate adding and deleting message properties in a COBOL CICS program. The program is simple:

1) After being triggered, it retrieves the control message that drives the operation. The control message holds the number of messages to be copied, the source queue, the target queue, and an optional message property value.
2) It reads a specified number of messages from the source queue, applies a message property and writes each message to the target queue. The message property name is COPY_PROP. The default value for this is “DEFAULT PROP”.
3) It deletes the property for the copied messages and creates the property for the status message. The status property name is (predictably) STATUS_PROP. The value is “TEST MESSAGE PROPERTY”.
4) Finally, it writes the status message, closes all the queues, and terminates.

This document assumes the reader is somewhat familiar with WMQ, CICS, and COBOL. The test samples make use of a Message broker SupportPac, IP13: WebSphere Business Integration Broker - Sniff test and Performance on z/OS. While technically a WMB SupportPac, this has been very useful for testing WMQ as well. This SupportPac may be found at:

http://www-01.ibm.com/support/docview.wss?rs=171&uid=swg24006892&loc=en_US&cs=utf-8&lang=en

Terms

Control Message – the message used to start the QCYP transaction. For QCYP the message contains, in comma delimited format:

1. The number of messages to be copied
2. The source queue
3. The target queue
4. The value to be used for the copy message property (optional)

Trigger message – this is the message passed to the processing program when the transaction is triggered, by placing the control message in the queue. A complete description of the trigger message is documented in the WebSphere MQ InfoCenter, or for those have the older MQ manuals in the Application Programming Reference (publication number SC34-6940).

QCOPYPRP

Program Description

The QCOPYPRP program is executed from the QCYP transaction. It copies messages from one queue to another applying a message property to each message. It is started by a comma delimited control message which triggers the transaction. It also uses information from the WMQ process object.

Inputs

The Copy Control message

The Copy Control message is a free form area; commas (',') are used to delimit the fields. A field can be omitted by including a blank and a comma (' ,') in it's place. If a field is omitted, a default value is supplied by the program. The data is broken up into the following fields:

```
01  COPY-CONTROL-MESSAGE.
    05  COPY-CONTROL            PIC 9(06) VALUE 1.
    05  SOURCE-QUEUE            PIC X(48) VALUE SPACES.
    05  TARGET-QUEUE            PIC X(48) VALUE SPACES.
    05  COPY-MESSAGE-PROP       PIC X(25) VALUE SPACES.
```

The fields are used as follows:

- COPY-CONTROL – The number of messages to copy from the source queue to the target queue. This can be a range of 1-99999 messages.
- SOURCE-QUEUE – The source of the messages to be copied. If not supplied, the default value is ‘QCOPYPRP.SOURCE.QUEUE'.
- TARGET-QUEUE – The target for the copied messages. If not supplied, the default value is ‘QCOPYPRP.TARGET.QUEUE'.
- COPY-MESSAGE-PROP – The property value to be applied to the copied messages. If not supplied, this defaults to ‘DEFAULT PROP’. The property name is always ‘COPY_PROP’.

Sample QCOPYPRP Copy Control Message

000003,QCOPYPRP.QCPY.INPUT.QUEUE1,QCOPYPRP.QCPY.OUTPUT.QUEUE1,TESTTHIS

The values were assigned as follows:

Field name	Value
COPY-CONTROL	3
SOURCE-QUEUE	QCOPYPRP.QCPY.INPUT.QUEUE1
TARGET-QUEUE	QCOPYPRP.QCPY.OUTPUT.QUEUE1
COPY-MESSAGE-PROP	TESTTHIS

The Trigger Message

Contains data fields that are used as follows:

- MQTM-QNAME – the name of the copy control queue, in the sample delivered it is 'QCOPYPRP.CONTROL.QUEUE'
- MQTM-ENVDATA – this is taken from the process definition, and may be used to supply the status queue name (see Outputs). If not supplied on the process definition, this defaults to 'QCOPYPRP.STATUS.QUEUE'.
- MQTM-USERDATA – if present this provides the message wait value for getting messages from the source queue.

Outputs

The Status Message

The status message has the follow layout:

```
01  STATUS-MESSAGE.
    05  FILLER                  PIC X(20)
                VALUE 'MESSAGES COPIED  = '.
    05  SM-NUMBER               PIC 9(6) VALUE ZEROS.
    05  FILLER                  PIC X(20)
                VALUE ' FROM QUEUE = '.
    05  SM-SOURCE-QUEUE         PIC X(48) VALUE SPACES.
    05  FILLER                  PIC X(20)
                VALUE ' TO QUEUE = '.
    05  SM-TARGET-QUEUE         PIC X(48) VALUE SPACES.
```

The fields are used as follows:

- SM-NUMBER is the total number of messages copied.
- SM-SOURCE-QUEUE – the source queue
- SM-TARGET-QUEUE – the target queue.

The copied messages

Each message is copied with the "COPY_PROP" message property added.

QCOPYPRP Program Flow

1. The QCYP transaction is triggered.
2. The control queue is opened.
3. Publication control message is read.
4. Control message is parsed into the controlling fields.
5. The source queue is opened.
6. The target queue is opened.
7. The message handle is created
8. The message property for the copied messages is setup.
9. In a loop, messages are read from the source queue and written to the target queue with the message property.
10. The message property is deleted.
11. The status message property is set up.
12. The status message is built.
13. The status queue opened and the status message is put.
14. All queues are closed.
15. Control is returned to CICS.

For this sample program, if the call to MQ fails the transaction will abend. The abend codes and their meanings are:

- QCP1 – The open of the control queue failed
- QCP2 – The open of the target queue failed
- QCP3 – The MQGET of the Copy Control message failed
- QCP4 – The open of the status queue failed
- QCP5 – The open of the source queue failed
- QCMH – The create message handle request failed
- QSMP – The set message property request failed
- QDMP – The delete message property request failed

The Message Properties code

The COBOL code to manipulate WebSphere MQ message properties is straightforward; it uses three of the WMQ V7 verbs:

1. Create Message Handle – this verb sets up the association between message properties the program defines and messages that are put. This is required before messages properties can be used by the application. For additional information, please see:
   http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/fr40140_.htm

   The sample code is shown:

```
*
*    CREATE A MESSAGE HANDLE TO STORE THE COPY MESSAGE PROPERTY
*
      CALL 'MQCRTMH' USING HCONN
                           MQM-CRT-MSG-HNDLE-OPTIONS
                           MESSAGE-HANDLE
                           COMPCODE
                           REASON.
```

2. Set Message Property – this defines message properties that can be associated with messages that are put. For additional information please see:
   http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/fr40770_.htm

   The sample code setting up the property:

```
*
*    SETUP COPY MESSAGE PROPERTIES
*
     IF COPY-MESSAGE-PROP EQUAL TO SPACES
        MOVE 'DEFAULT PROP' TO COPY-MESSAGE-PROP.
     MOVE COPY-MESSAGE-PROP TO COPY-PROP.
     COMPUTE MQSMPO-OPTIONS = MQSMPO-SET-FIRST.
     SET MQCHARV-VSPTR TO ADDRESS OF COPY-PROP-NAME
     MOVE LENGTH OF COPY-PROP-NAME  TO
                           MQCHARV-VSBUFSIZE.
     MOVE LENGTH OF COPY-PROP-NAME TO
                           MQCHARV-VSLENGTH.
     COMPUTE MQPD-CONTEXT = MQPD-USER-CONTEXT.
     COMPUTE PROPERTY-TYPE = MQTYPE-STRING.
     MOVE LENGTH OF COPY-MESSAGE-PROPERTY  TO
                           PROPERTY-VALUE-LENGTH.
*
```


The MQ call:

```
*
*    CREATE MESSAGE PROPERTIES
*
     CALL 'MQSETMP' USING HCONN
                          MESSAGE-HANDLE
                          MQM-SET-MESSAGE-OPTIONS
                          MQM-SET-PROPERTY-NAME
                          MQM-SET-PROPERTY-DESCRIPTOR
                          PROPERTY-TYPE
                          PROPERTY-VALUE-LENGTH
                          COPY-PROP
                          COMPCODE
                          REASON.
```

3. Delete Message Property – this deletes a previously created message property, so it will no longer be included with any put messages. For additional information please see:

   http://publib.boulder.ibm.com/infocenter/wmqv7/v7r1/topic/com.ibm.mq.doc/fr25400_.htm

   The Delete Message property sample

```
*
*    DELETE COPY  MESSAGE PROPERTY
*
     CALL 'MQDLTMP' USING HCONN
                          MESSAGE-HANDLE
                          MQM-DELETE-MESSPROP-OPTIONS
                          MQM-SET-PROPERTY-NAME
                          COMPCODE
                          REASON.
```

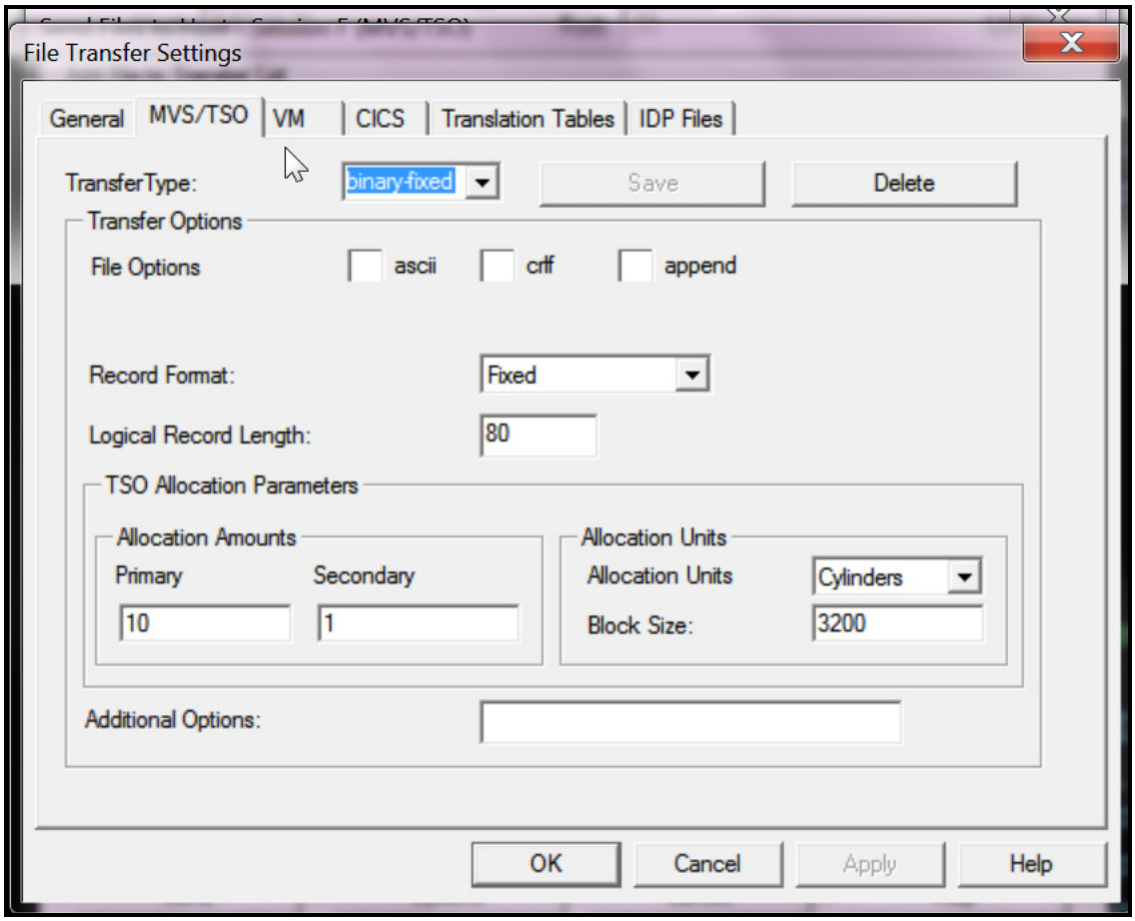

Installing the Sample

Uploading the sample file

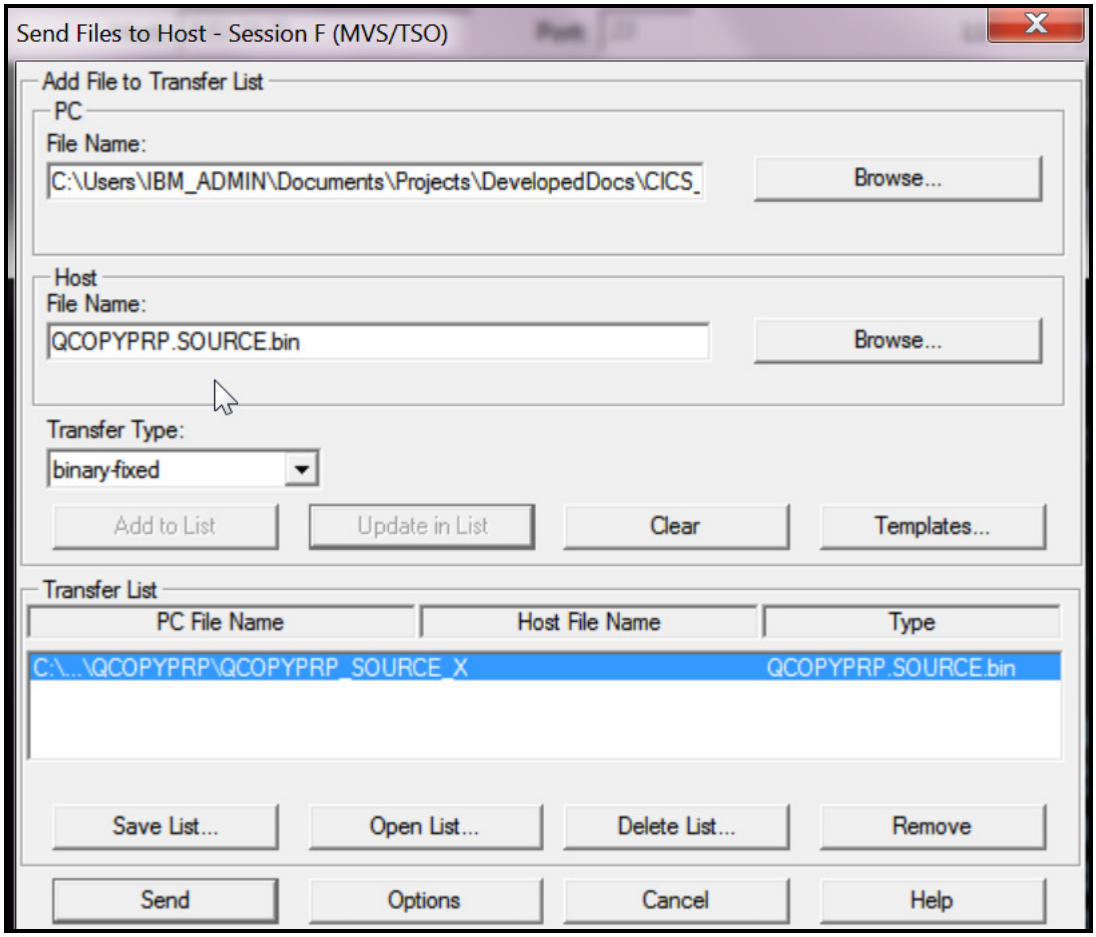
The samples file contains the source for the copy with properties program, the WMQ definitions, the CICS definitions, JCL for the test, sample data, and an edit exec that can be used to alter the '++' variables in the definitions and JCL. The file must be uploaded to z/OS in binary fixed length record format and then received to create the PDS.

And example of the upload, done via PCOM, and the TSO receive are shown below.

1) Upload the QCOPYPRP_SOURCE_XMIT.BIN file. Make sure that the sequential file is a fixed length 80 byte file, and that you use the binary format. The transfer type used in the ATS test was:



2) The PCOM upload looked as follows:



3) Once the upload has been completed, then use the TSO receive command to rebuild the source PDS. The receive command is entered from a TSO ready prompt or from the TSO command panel as shown.

```
receive indsname('elkinsc.qcopyprp.source.bin')
```

4) When prompted either hit the enter key to accept the default names or enter the DSNAME('your.source.dataset') command to suite your naming conventions.

5) Display the contents of the created PDS, it should look as illustrated.

```
BROWSE            ELKINSC.QCOPYPRP.SOURCE                  Row 00001 of 00009
Command ===> _                                               Scroll ===> CSR
           Name     Prompt       Size   Created         Changed          ID
_________ CICSDEFS                 19  2013/08/07  2013/08/23 13:38:52  ELKINSC
_________ COPYTSTM                  1  2013/08/08  2013/08/24 10:40:55  ELKINSC
_________ LOADMSGS                 10  2013/08/08  2013/08/08 12:30:19  ELKINSC
_________ QCOPYPRP                625  2013/08/23  2013/08/23 15:05:33  ELKINSC
_________ QCYPEDIT                 29  2013/08/07  2013/08/23 15:34:45  ELKINSC
_________ QCYPMSG                  10  2013/08/23  2013/08/23 15:49:43  ELKINSC
_________ QCYPPROC                  9  2013/08/23  2013/08/23 14:59:40  ELKINSC
_________ QCYPQUES                189  2013/08/23  2013/08/23 14:58:25  ELKINSC
_________ QCYPTEST                 99  2013/08/23  2013/08/24 11:14:42  ELKINSC
```

6) Repeat the upload and receive steps for the load dataset. The load library should contain only one member:

```
BROWSE            ELKINSC.QCOPYPRP.LOAD                    Row 00001 of 00001
Command ===> _                                               Scroll ===> CSR
           Name     Prompt       Alias-of      Size      TTR     AC   AM   RM
_________ QCOPYPRP                          00007918   000004   00    31  ANY
```


Customizing the Sample

Edit the REXX edit file

For convenience a REXX exec has been included that has change commands to tailor all the '++' variables used in the other samples to those suitable for your environment. These steps describe editing and using the REXX exec to tailor the members.

1) Open the QCYPEDIT member in edit mode. It should looks as follows:

```
000001 ISREDIT MACRO NOPROCESS
000002 ADDRESS ISREDIT
000003 /* **** CICS DEFINITIONS CHANGES ****  */
000004 "CHANGE '++QML0GRP++' 'CICSGRP' ALL"
000005 "change '++QCYP++' 'QCYP' all"
000006 "change '++QCOPYPRP++' 'QCOPYPRP' all"
000007 "change '++USER++' 'ELKINSC' all"
000008 /* **** OEMPUTX CHANGES ****  */
000009 "change '++IP13.LOADLIB++' 'SYS1.MQM.IP13.LOADLIB' all"
000010 "change '++WMQHLQ++' 'SYS1.MQV701' all"
000011 "change '++THIS.PDSNAME++' 'QCYP.INSTALL.PDS' all"
000012 "change '++QMGR++' 'CSQ1' all"
000013 "change '++DOC.SUMMARY++' 'QCYP.IP13.DOC.SUMARY' all"
000014 "change '++DB2.NAME++' 'DSNA' all"
000015 "change '++DOC.SUMMARY++' 'QCYP.IP13.DOC.SUMARY' all"
000016 "change '++IP13.TEMP++' 'QCYP.IP13.TEMP' all"
000017 "change '++QCYP.CONTROL.QUEUE++' 'QCYP.CONTROL.QUEUE' all"
000018 "change '++QCYP.INPUT.QUEUE++' 'QCYP.INPUT.QUEUE' all"
```

2) Only change the sample values, those on the right. If the ++ variables are changed, in this member, they will not be changed in the other members. As an example, the value 'QML0GRP' will be changed to 'CICSGRP'. You would need to change that to the RDO group selected for this sample.

   Note that some of the variables are repeated because these values are used in multiple members.

3) Once the values are changed, activate the library using the following command:

   ALTLIB ACTIVATE APPLICATION(EXEC) DA('your.source.dataset')

4) Apply the edits by entering the 'QCYPEDIT' command to alter the ++ variables to those valid in your environment.

5) As an example, the CICSDEFs member looks as shown before editing:

```
EDIT       ELKINSC.QCOPYPRP.SOURCE(CICSDEFS) - 01.02       Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ********************************
000001  DEFINE PROGRAM(++QCOPYPRP++) GROUP(++QML0GRP++)
000002  DESCRIPTION(SAMPLE WMQ COPY WIH PROPERTIES)
000003         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000004         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000005         EXECKEY(USER) CONCURRENCY(THREADSAFE) API(CICSAPI) DYNAMIC(NO)
000006         EXECUTIONSET(FULLAPI) JVM(NO) JVMPROFILE(DFHJVMPR)
000007         DEFINETIME(11/08/30 05:45:17) CHANGETIME(11/08/30 05:45:46)
000008         CHANGEUSRID(++USER++) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
000009  DEFINE TRANSACTION(++QCYP++) GROUP(++QML0GRP++)
000010  DESCRIPTION(SAMPLE WMQ COPY WITH PROPERTIES TRANSACTION)
000011         PROGRAM(++QCOPYPRP++) TWASIZE(0) PROFILE(DFHCICST) STATUS(ENABLE
000012         TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
000013         RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
000014         ROUTABLE(NO) PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO)
000015         RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
000016         CONFDATA(NO) OTSTIMEOUT(NO) ACTION(BACKOUT) WAIT(YES)
000017         WAITTIME(0,0,0) RESSEC(NO) CMDSEC(NO)
000018         DEFINETIME(11/08/30 05:43:04) CHANGETIME(11/08/30 05:43:04)
000019         CHANGEUSRID(++USER++) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
```

6) Following the execution of qcypedit, the member has been updated as shown:

```
EDIT       ELKINSC.QCOPYPRP.SOURCE(CICSDEFS) - 01.03       Columns 00001 00072
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ********************************
==CHG>  DEFINE PROGRAM(QCOPYPRP) GROUP(CICSGRP)
000002  DESCRIPTION(SAMPLE WMQ COPY WIH PROPERTIES)
000003         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000004         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000005         EXECKEY(USER) CONCURRENCY(THREADSAFE) API(CICSAPI) DYNAMIC(NO)
000006         EXECUTIONSET(FULLAPI) JVM(NO) JVMPROFILE(DFHJVMPR)
000007         DEFINETIME(11/08/30 05:45:17) CHANGETIME(11/08/30 05:45:46)
==CHG>         CHANGEUSRID(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
==CHG>  DEFINE TRANSACTION(QCYP) GROUP(CICSGRP)
000010  DESCRIPTION(SAMPLE WMQ COPY WITH PROPERTIES TRANSACTION)
==CHG>         PROGRAM(QCOPYPRP) TWASIZE(0) PROFILE(DFHCICST) STATUS(ENABLED)
000012         TASKDATALOC(ANY) TASKDATAKEY(USER) STORAGECLEAR(NO)
000013         RUNAWAY(SYSTEM) SHUTDOWN(DISABLED) ISOLATE(YES) DYNAMIC(NO)
000014         ROUTABLE(NO) PRIORITY(1) TRANCLASS(DFHTCL00) DTIMOUT(NO)
000015         RESTART(NO) SPURGE(NO) TPURGE(NO) DUMP(YES) TRACE(YES)
000016         CONFDATA(NO) OTSTIMEOUT(NO) ACTION(BACKOUT) WAIT(YES)
000017         WAITTIME(0,0,0) RESSEC(NO) CMDSEC(NO)
000018         DEFINETIME(11/08/30 05:43:04) CHANGETIME(11/08/30 05:43:04)
==CHG>         CHANGEUSRID(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
```

7) Repeat the edit process for all members, except qcypedit itself.

CICS Definitions

There are two CICS definitions required; one transaction (QCYP) and one program QCOPYPRP. The definitions are in the CICSDEFS member, and may be used with the DFHCSDUP utility to define the resources. Alternatively, RDO (the CEDA CICS transaction) may be used to define the resources.

For information on the DFHCSDUP utility, please see:

http://publib.boulder.ibm.com/infocenter/cicsts/v4r1/index.jsp?topic=%2Fcom.ibm.cics.ts.resourcedefinition.doc%2Fcsdup%2Fdfhcsdup.html

WMQ Definitions

The WMQ definitions are in members in the source library, and should be edited to ensure compliance with your standards.

Process Definitions

The member QCYPPROC is the process definition used to trigger the QCYP transaction. The PDS member may be used as input to he CSQUTIL program to define the processes in batch mode, or the objects can be defined online via the WMQ ISPF panels or the MQ Explorer.

Queue Definitions

The members QCYPQUES contain the queue definitions required to implement the sample. The PDS members may be used as input to he CSQUTIL program to define the processes in batch mode, or the objects can be defined online via the WMQ ISPF panels or the MQ Explorer.

Testing the program

The following steps may be used to test the QCOPYPRP program.

1. If not already installed, install the IP13 SupportPac. It can be found at:

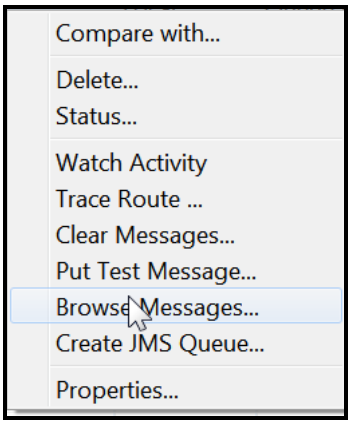
   http://www-01.ibm.com/support/docview.wss?uid=swg24006892

   While technically a WebSphere Message Broker (IBM Integration Bus as it is now known) SupportPac, this is very useful for testing WebSphere MQ programs.
2. Define the resources to the queue manager you will use for this test. The members contain listed below the generic MQ object definitions. You can modify the samples (using the sample edit REXX) and use CSQUTIL to create the definitions, or you can create the definitions via the explorer or ISPF panels. The object definition members are:
   a. QCYPPROC – the process definition to trigger the QCYP transaction
   b. QCYPQUES – the sample queue definitions.
3. Define the CICS resources needed for the test. The sample program nad transaction definition are in the CICSDEFS member of the source library.
4. Add the sample program to your CICS environment. This can be done by one of the following:
   a. Compiling and linking the sample program, QCOPYPRP, into a load library already defined to the CICS region.
   b. Copying the load module into a load library already defined to the CICS region.
   c. Adding the load library delivered with this sample to the RPL list of the test CICS region.
5. Modify the test JCL, which is in member QCYPTEST in the delivered source file.
6. Run the test JCL. At the end of the test the queue depths should look as shown:

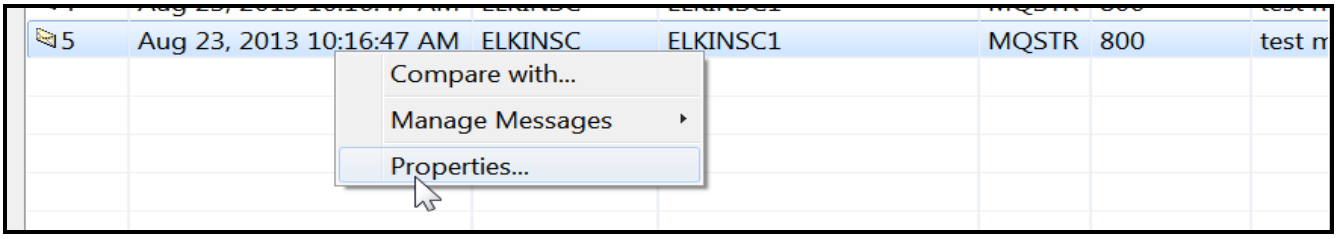
Filter: QCOPYPRP

Queue name	Queue type	QSG disp...	Open input c...	Op...	Current queue ...	Put messages	Get messages
QCOPYPRP.CONTROL.QUEUE	Local	Queue m...	0	0	0	Allowed	Allowed
QCOPYPRP.INPUT.QUEUE	Local	Queue m...	0	0	5	Allowed	Allowed
QCOPYPRP.OUTPUT.QUEUE	Local	Queue m...	0	0	5	Allowed	Allowed
QCOPYPRP.STATUS.QUEUE	Local	Queue m...	0	0	1	Allowed	Allowed

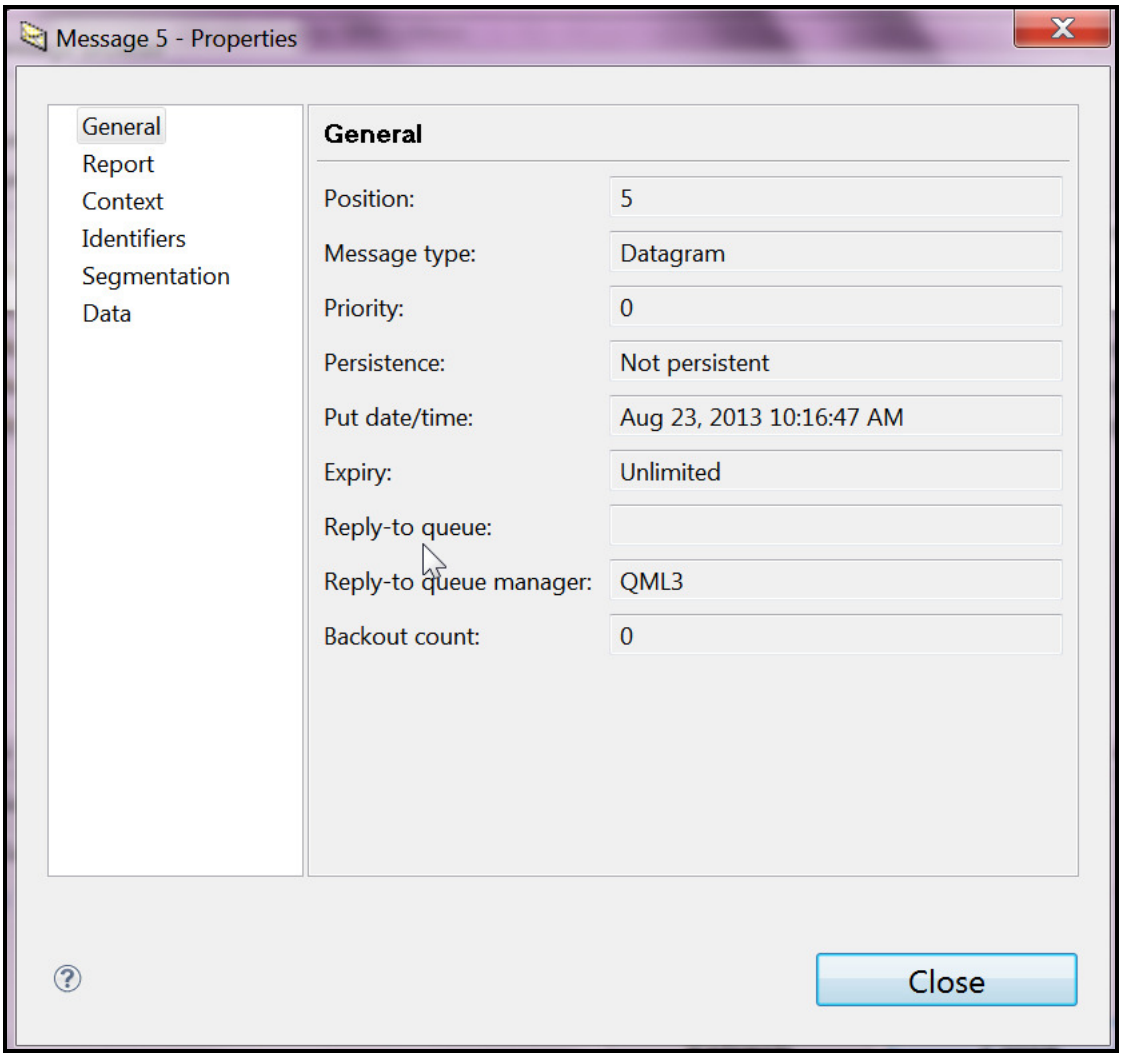
   The input queue should have 5 message, the output queue has 5 messages, and the status queue should have one.
7. To verify the results, first browse the input queue. These messages have been placed there by the OEMPUTX program. They should not have any message properties. To use the MQ explorer to browse the messages{
   a. Right click on the input queue, and select ‘Browse messages;



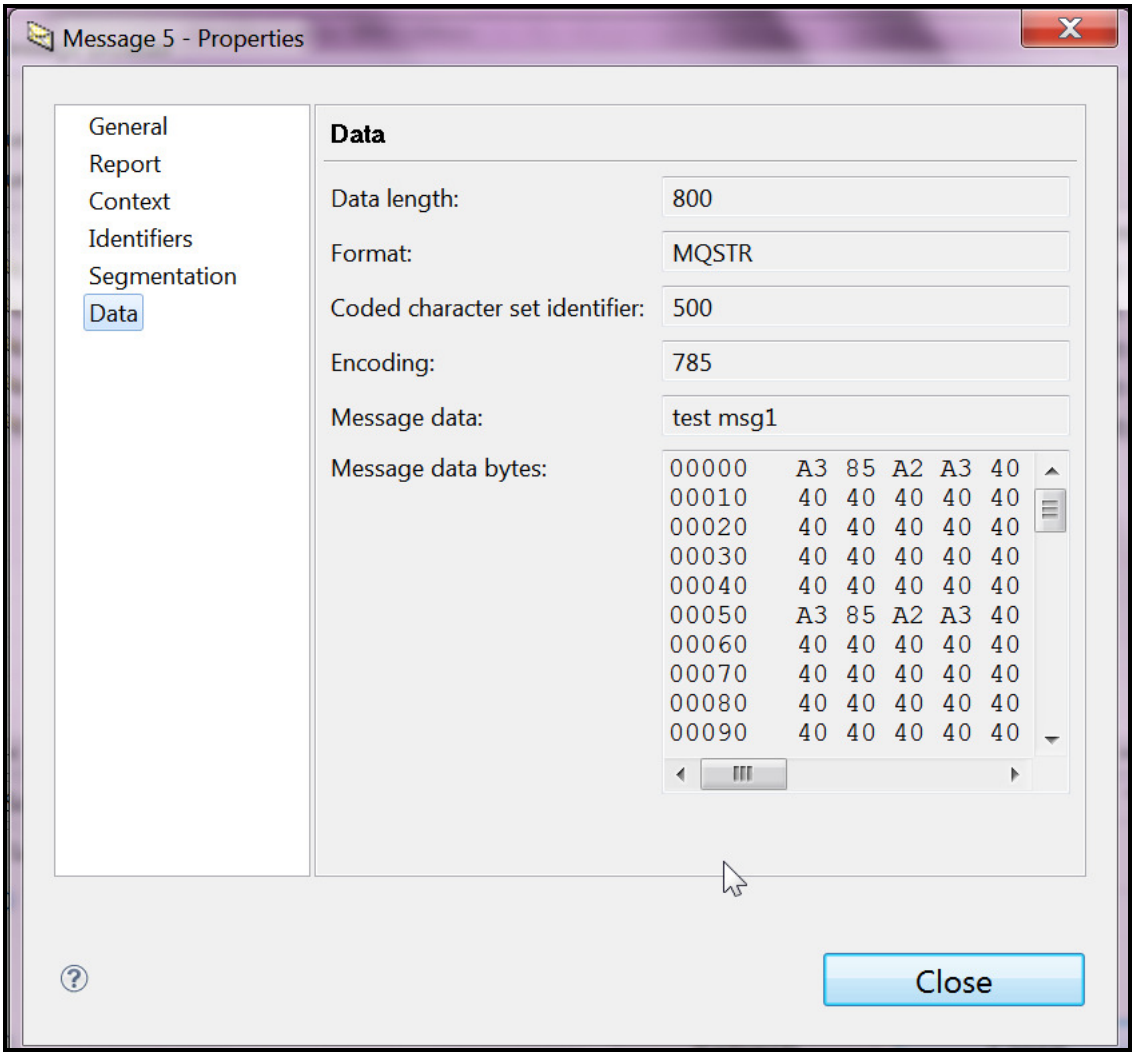
8. The messages on the queue will be displayed. Right click on any of the messages and select Properties as shown.



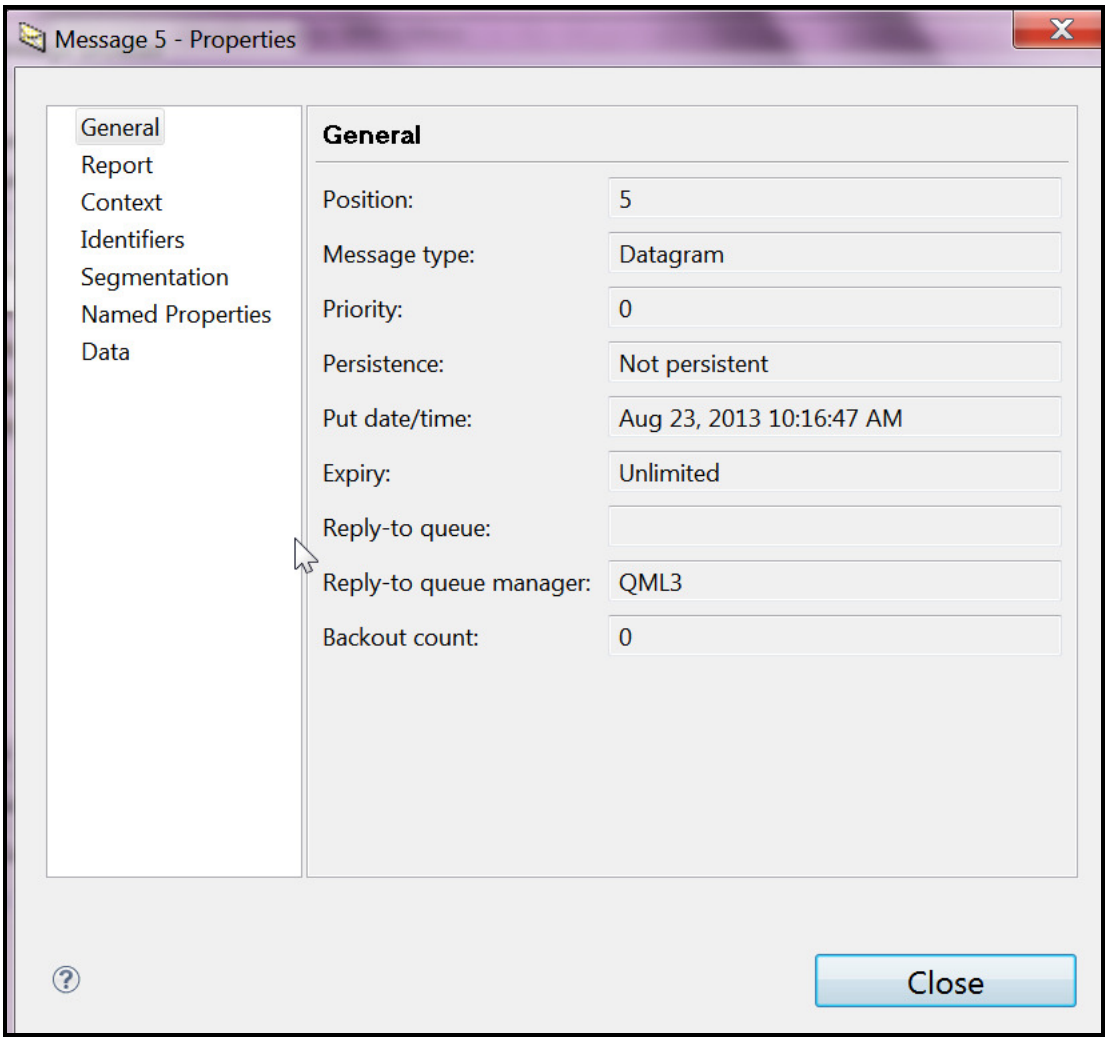
b. Note that there are no ‘Named properties’ associated with these messages, as that tab does not show up on this display.



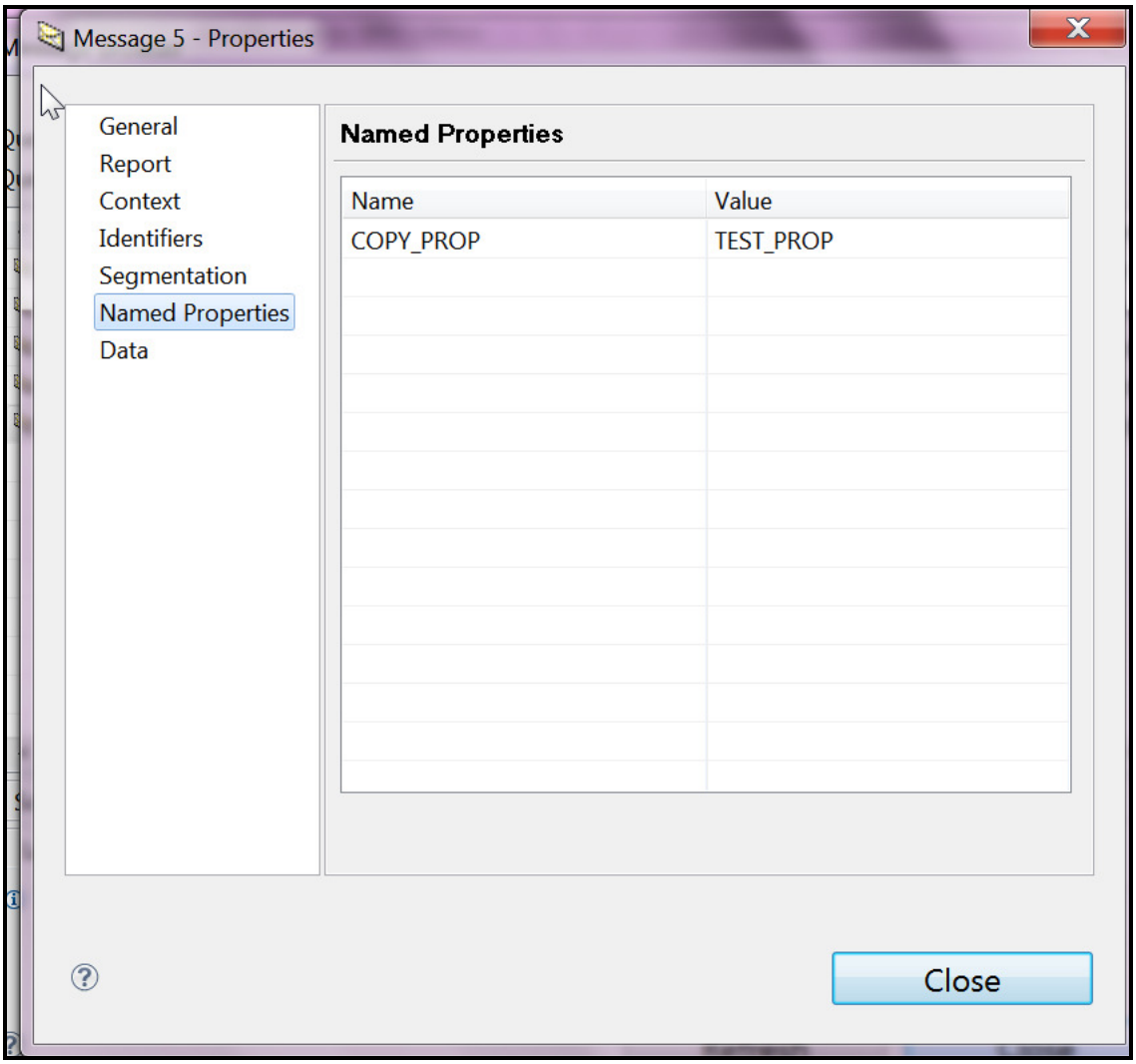
c. Selecting the Data folder will display the message contents.



9. Next browse the output queue to verify that a property has been added to the messages.
    1. Right click on the output queue, and select ‘Browse messages’.
    2. Right click on one of the messages. In the output queue, you should see he ‘Named properties’ folder as shown:

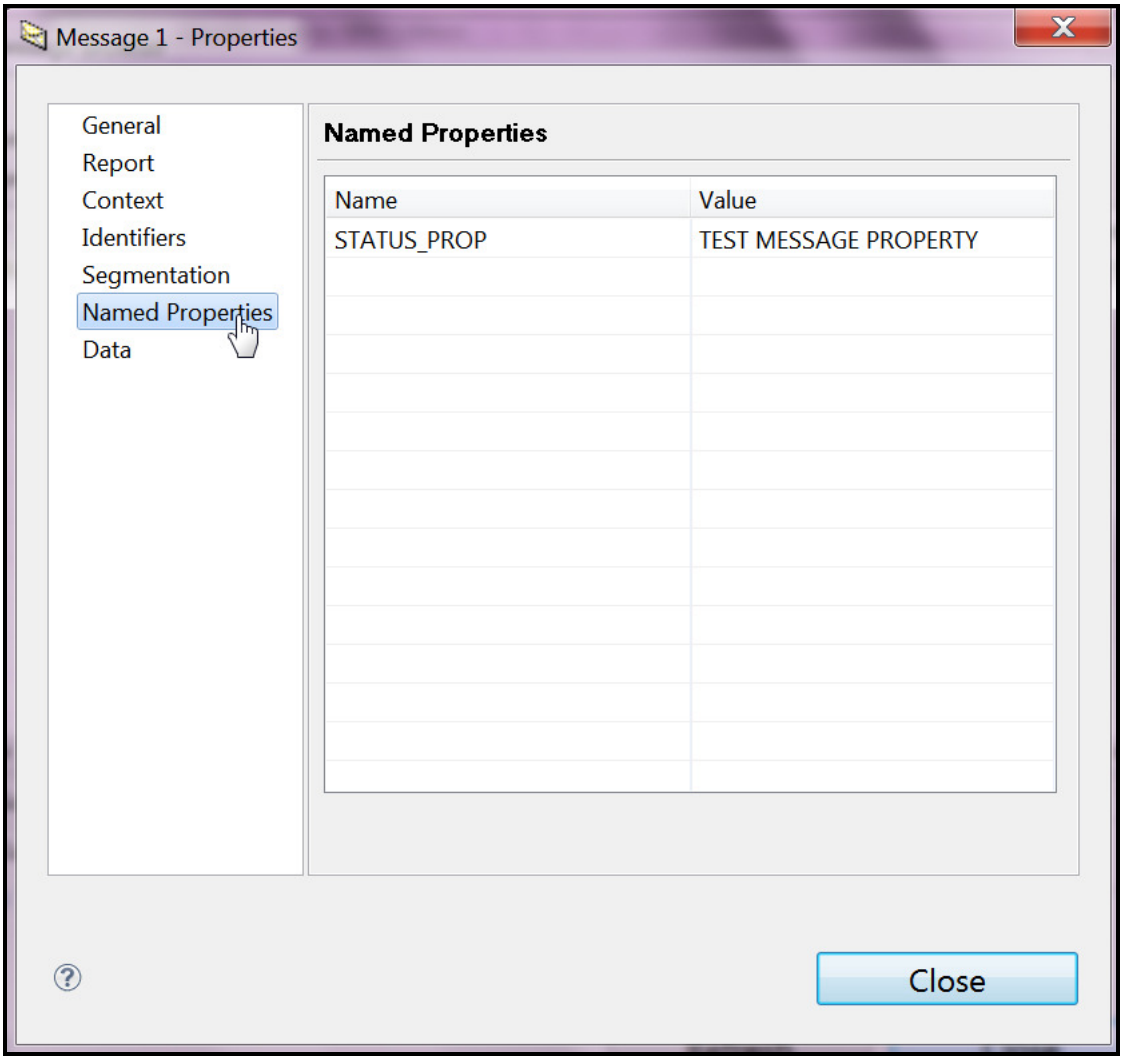


3. Select the ‘Named Properties’ folder, and the message property added on the copy is displayed.

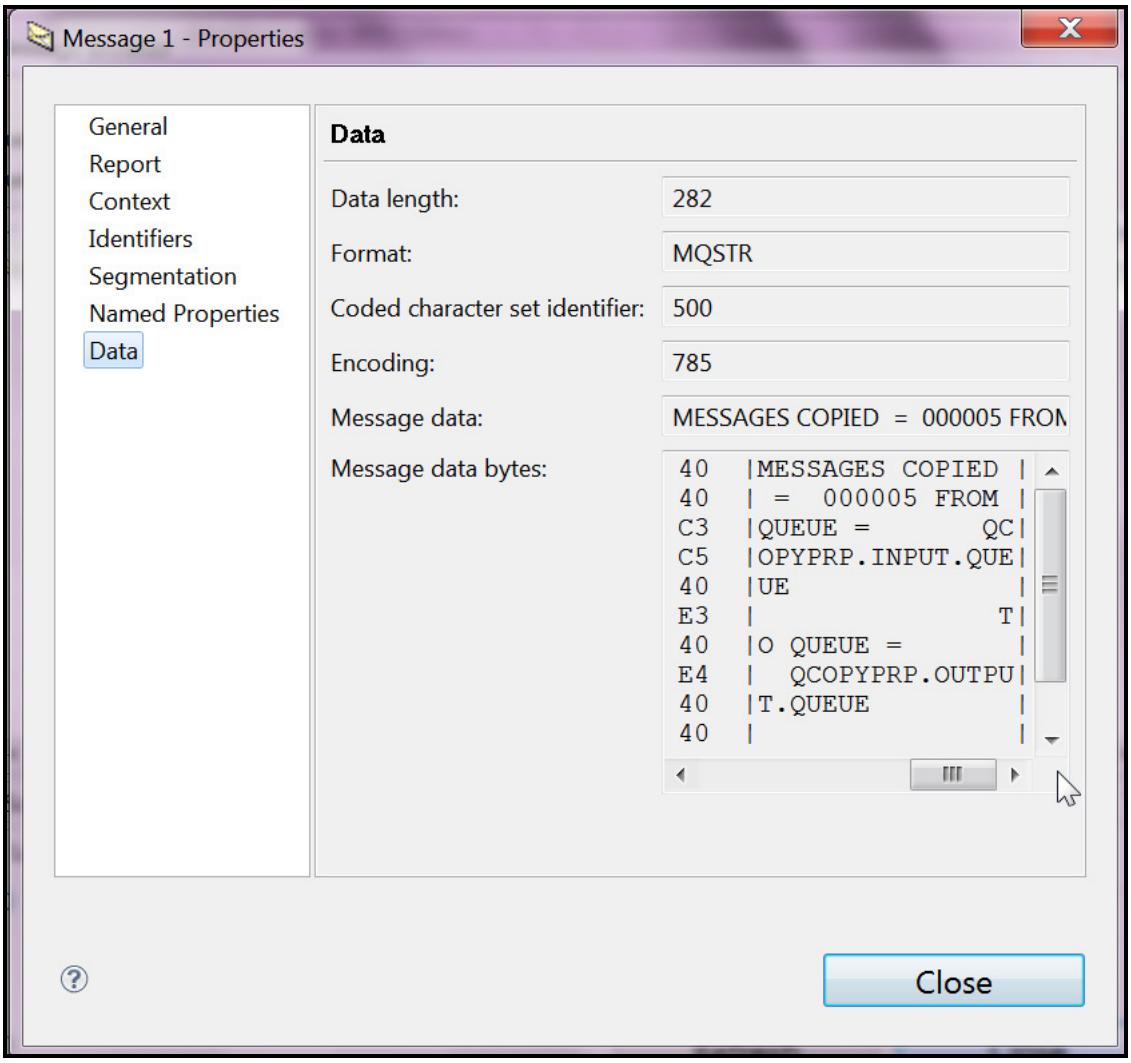


d. The Data folder should look like the data folder from the input queue.
e. Close the panels back to the queue list.

10. Finally, verify that the status message was created correctly and has a property associated.
    a. Right click on the status queue, and select ‘Browse messages’.
    b. Right click on the message. The ‘Named Properties’ folder should appear in the list, as it did on the output queue.
    c. On the status message the named property should look as shown:



f. Select the data folder, and the message contents should look something like this (queue names may be different):



Congratulations!

Acknowledgments:

The authors would like to thank the following people for their assistance
Mark Taylor
Shalawn King
Kenishia Sapp
Chris Griego