QSU2 & QPU2

Sample COBOL CICS WMQ Publication and Subscription Transactions
Topic Strings with embedded blanks

The IBM ATS WebSphere MQ team:

Lyn Elkins – elkinsc@us.ibm.com
Mitch Johnson – mitchj@us.ibm.com
Eugene Kuehlthau - ekuehlth@us.ibm.com
Ed Zeilnhofer - edz@us.ibm.com

QPU2 & QSU2 – Sample COBOL WMQ CICS Pub/Sub Transactions

Introduction....4
Terms....5
QPU2 – Sample CICS publication transaction 6
Output:....7
The publication status message, which has the following layout:....7
QSU2 – Sample CICS transaction to subscribe to publications....9
Sample Subscription control message: 10
Output:....11
The subscription status message, which has the following layout:....11
Installing the samples....14
Process Definitions....21
Queue Definitions....21
Topic Definition....21
Testing the Sample Programs....22
Testing Results – PUTXTST1....22
Testing Results – PUTXTST2....24
Testing Results – PUTXTST3....26
Appendix A – CICS RDO definition sample....28
QML0GRP Installation....30
Appendix B – Sample Compile and Link JCL 33

Introduction

This document describes the sample COBOL MQ CICS transactions called QSU2 and QPU2. These transactions use the publish/subscribe feature of WMQ V7 and requires CICS 3.2 or above.

Please note that the following PTFs 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 transactions are single purpose – all QSU2 does it subscribe to a topic and pull messages from the targeted queue, all QPU2 does is publish messages to a specified topic. These transactions are almost exactly like their predecessors, QPUB and QSUB transactions with one exception, these will support topic strings that have embedded blanks. This is enabled by supplying a delimiter for the topic string. The delimiter is a single byte, and can be any character except a comma (‘,’). The delimiter can be low values (x’00’). The first instance of the delimiter terminates the topic string, any characters following the delimiter are ignored. While there is nothing preventing a space from being specified as the delimiter, be aware that the first space in the topic string will be the termination point for the string.

MQ messages and attributes of the process definitions are used to provide the necessary information to create the publication, the subscription, and to report on the status of the but the publication and subscription requests.

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 QPU2 and QSU2 transactions. For QPU2 the message contains:

1. The number of messages to be published,
2. The MQ Topic Object (optional)
3. The Topic String delimiter (optional)
4. The Topic String (optional)

For QSU2 the control message contains:

1. The number of messages to get
2. The subscription queue
3. The MQ Topic object (optional)
4. The topic string delimiter (optional)
5. The MQ Topic string (optional)
6. A subscription correlation ID (optional)
7. A subscription name (optional)

Resolved topic – the resolved topic is the complete topic string used for publications. It can be composed of :

1. The topic string attribute from the defined topic object specified in the control message, or
2. The topic string supplied in the control message when no topic object has been specified, or
3. The concatenation of the topic string attribute of the topic object from the control message followed by a slash (‘/’) and the topic string from the control message.

In the sample program, the resolved topic is limited to 400 characters.

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).

.

QPU2 – Sample CICS publication transaction

Description:

The QPU2 transaction executes the QPU2CBL program; it publishes messages besed on the provided topic object, topic string or a combination of the two fields. It is started by a formatted 'control message' which triggers the transaction. It also uses information from the WMQ process object.

Inputs:

The Publication Control message.

The Publication 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. The data is broken up into the following fields:

```
01  PUBLISH-PARAMETERS.
    05  NUMBER-OF-PUBS          PIC 9(5)   VALUE ZEROS.
    05  TOPIC-OBJECT            PIC X(48)  VALUE SPACES.
    05  TOPIC-DELIM             PIC X(1)   VALUE SPACES.
    05  TOPIC-STRING            PIC X(200) VALUE SPACES.
```

The fields are used as follows:

- NUMBER-OF-PUBS – The number of publications to be made to the resolved topic object (combination or the topic object and topic string). This can be a range of 1-99999 messages.
- TOPIC-OBJECT is the MQ defined topic object. The field is optional, if supplied, the topic string from the topic object definition is used as part (or the whole of) the resolved topic.
- TOPIC-DELIM is the delimiter that terminates the TOPIC-STRING field. It an optional field, and should be used when the TOPIC-STRING has embedded blanks.
- TOPIC-STRING is an optional component of the resolved topic. If omitted and the TOPIC-OBJECT is also omitted, the SYSTEM.DEFAULT.TOPIC is used for subscription. If it is present, and the TOPIC-OBJECT is omitted, this value is used as the resolved topic for publication. If it is present and the TOPIC-OBJECT is also present, the resolved topic is created by WMQ from the topic string defined on the topic object, a concatenated slash to indicate hierarchy, and the topic string.

Sample Publication Control Message

2,QSU2.TOPIC,(,TEST/LEVEL 1 (

The values were assigned as follows:

Field name	Value
NUMBER-OF-PUBS	2
TOPIC-OBJECT	QSU2.TOPIC
TOPIC-DELIM	(
TOPIC-STRING	TEST/LEVEL 1 Note that the delimiter is not part of the string

The Trigger message contains data fields that are used as follows:

- MQTM-QNAME – the name of the publication control queue, in the sample delivered it is 'QPU2.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 'QPU2.STATUS.QUEUE'.
- MQTM-USERDATA – this is taken from the process definition and may be used to supply a message body for the publications.

Output:

The publication status message, which has the following layout***:***

```
01  STATUS-MESSAGE.
     05  FILLER                  PIC X(20)
                 VALUE 'PUBLICATIONS MADE= '.
     05  SM-NUMBER               PIC 99999 VALUE ZEROS.
     05  FILLER                  PIC X(20)
                 VALUE ' FOR TOPIC = '.
     05  SM-TOPIC                PIC X(48) VALUE SPACES.
     05  FILLER                  PIC X(20)
                 VALUE ' RESOLVED  = '.
     05  SM-RESOLVED             PIC X(400) VALUE SPACES.
```

SM-NUMBER is the total number of messages published.
SM-TOPIC – the topic object, if provided, from the publication control message.
SM-RESOLVED – this is the resolved topic string used for the publication.

QPU2CBL Program Flow:

1) The QPU2 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 topic is opened.
6) Messages are published in a loop, until the control count has been reached or the wait time.
7) The status message is built.
8) The status queue opened and the status message is put.
9) All queues are closed.
10) Control returns to CICS.

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

- QPU1 – The open of the Publication control queue failed
- QPU2 – The open of the TOPIC failed
- QPU3 – The MQGET of the Publication Control message failed
- QPU4 – The MQPUT of the status message failed
- QPU5 – The open of the status queue failed

QSU2 – Sample CICS transaction to subscribe to publications

Description:

The QSU2 transaction executes the QSU2CBL program; it subscribes to publications based on the provided topic object, topic string or a combination of the two fields. It is started by a formatted 'control message' which triggers the transaction. It also uses information from the WMQ process object.

Inputs:

The Subscription Control message.

The Subscription Control message is a free form area; commas are used to delimit the fields. The data is broken up into the following fields:

```
01  SUBSCRIPTION-PARMS.
    05  SUB-CONTROL              PIC 9(05)  VALUE 1.
    05  SUB-QUEUE                PIC X(48)  VALUE SPACES.
    05  TOPIC-OBJECT             PIC X(48)  VALUE SPACES.
    05  TOPIC-STR-DEL            PIC X(01)  VALUE SPACES.
    05  TOPIC-STRING             PIC X(200) VALUE SPACES.
    05  MATCH-CORRELID           PIC X(24)  VALUE SPACES.
    05  SUB-NAME                 PIC X(200) VALUE SPACES.
```

The fields are used as follows:

- SUB-CONTROL is the number of messages to be retrieved from the subscription queue. This can be a range of 00001-99999. This is a required field.
- SUB-QUEUE is the name of the subscription destination queue, where the publications are delivered. This sample does not allow the use of managed subscriptions, this is a required field. The maximum length is 48 characters.
- TOPIC-OBJECT is the MQ defined topic object. The field is optional, if supplied, the topic string from the topic object definition is used as part (or the whole of) the resolved topic.
- TOPIC-STR-DEL is the delimiter that terminates the TOPIC-STRING field. It an optional field, and should be used when the TOPIC-STRING has embedded blanks.

- TOPIC-STRING is an optional component of the resolved topic. If omitted and the TOPIC-OBJECT is also omitted, the SYSTEM.DEFAULT.TOPIC is used for subscription. If it is present, and the TOPIC-OBJECT is omitted, this value is used as the resolved topic for the subscription. If it is present and the TOPIC-OBJECT is also present, the resolved topic is created by WMQ from the topic string defined on the topic object, a concatenated slash to indicate hierarchy, and the topic string.
- MATCH-CORRELID is an optional field, if provided:
    - It sets a fixed correlation ID on all the publications delivered as a result of this subscription
    - Is used as a match option for the MQGETs.
- SUB-NAME is an optional field, which may be used to name the subscription.

Sample Subscription control message:

00005,QSU2.SUB.QUEUE,QPUB.TEST, ,LYN CORRELID1,SUBNAME-LYN

The values were assigned as follows:

Field name	Value
SUB-CONTROL	5
SUB-QUEUE	QSU2.SUB.QUEUE
TOPIC-OBJECT	QPUB.TEST
TOPIC-STR-DEL	space (note a space must be left between the commas to omit a parameter)
TOPIC-STRING	spaces (note a space must be left between the commas to omit a parameter)
MATCH-CORRELID	LYN CORRELID1
SUB-NAME	SUBNAME-LYN

The following table describes the resolved topic that will be used for the subscription. It assumes that the topic object named has a topic string that contains the name, though that is not a requirement.

TOPIC-OBJECT	TOPIC-STRING	Resolved topic
Blank	Blank	SYSTEM.DEFAULT.TOPIC
Fruit	Blank	Fruit
Cookware	Castiron/Skillet	Cookware/Castiron/Skillet
Blank	ShoeSale/Sneakers	ShoeSale/Sneakers

The Trigger message contains data fields that are used as follows:

- MQTM-QNAME – the name of the publication control queue, in the sample delivered it is 'QSU2.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 'QSU2.STATUS.QUEUE'.
- MQTM-USERDATA – this is taken from the process definition and may be used to supply a wait interval for the MQGET on the subscription queue.

Output:

The subscription status message, which has the following layout***:***

```
01  STATUS-MESSAGE.
     05  FILLER                  PIC X(20)
                 VALUE 'PUBLICATIONS MADE= '.
     05  SM-NUMBER               PIC 99999 VALUE ZEROS.
     05  FILLER                  PIC X(20)
                 VALUE ' FOR TOPIC = '.
     05  SM-TOPIC                PIC X(48) VALUE SPACES.
     05  FILLER                  PIC X(20)
                 VALUE ' RESOLVED  = '.
     05  SM-RESOLVED             PIC X(400) VALUE SPACES.
```

The fields are used as follows:

SM-NUMBER is the total number of messages retrieved during the subscription period.
SM-TOPIC – the topic object, if provided, from the subscription control message.
SM-RESOLVED – this is the resolved topic string used for the subscription.

If testing with a resolved topic string of more than 400 characters is required, the size can be adjusted in the program QPU2CBL and the program compiled and linked. The following fields may need to be altered to support a larger area:

- TOPIC-STR – 200 byte working storage field that has a copy of the input topic string from the subscription control message.
- RESOLVED-TOPIC-STR – 400 byte working storage field that is used to pass to WMQ to construct the resolved topic string from the topic object and topic string from the subscription control message.
- TOPIC-STRING – 200 byte working storage field that is part of the subscription control message.

- SM-RESOLVED - 400 byte working storage field that is part of the status message, it gives the resolved topic string returned from WMQ

QSU2CBL Program Flow:

11) The QSU2 transaction is triggered.

12) The control queue is opened.

13) Subscription control message is read.

14) Control message is parsed into the controlling fields.

15) The subscription request is built and submitted.

16) The subscription destination queue is opened.

17) Messages are read from the queue in a loop, until the control count has been reached or the wait time.

18) The status message is built.

19) The status queue opened and the status message is put.

20) All queues are closed.

21) Control returns to CICS.

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

- QSU1 – The open of the Subscription control queue failed
- QSU2 – The MQSUB, the subscription request itself, failed
- QSU3 – The MQGET of the Subscription Control message failed
- QSU4 – the open of the Status Queue failed
- QSU5 – The open of the Subscription destination queue failed

Important note: The subscription is not closed until the transaction ends, during testing you may see published messages on the subscription destination queue that have arrived after the queue was closed but before the transaction ended. This is because of the structure of the program and the nature of pub/sub. Using message expiration and clean up either within application or with another program or tool should be considered when creating pub/sub applications.

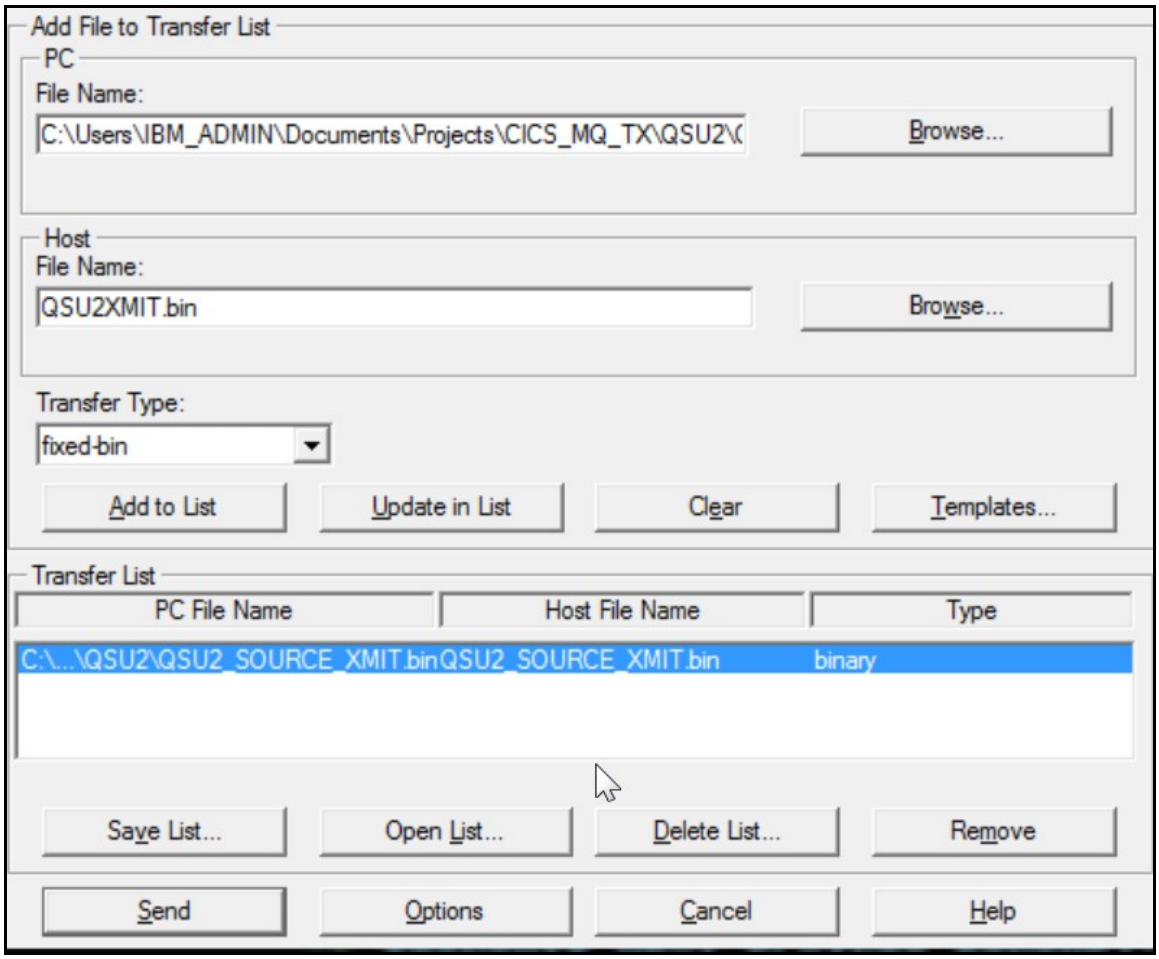
Installing the samples

Uploading the samples file

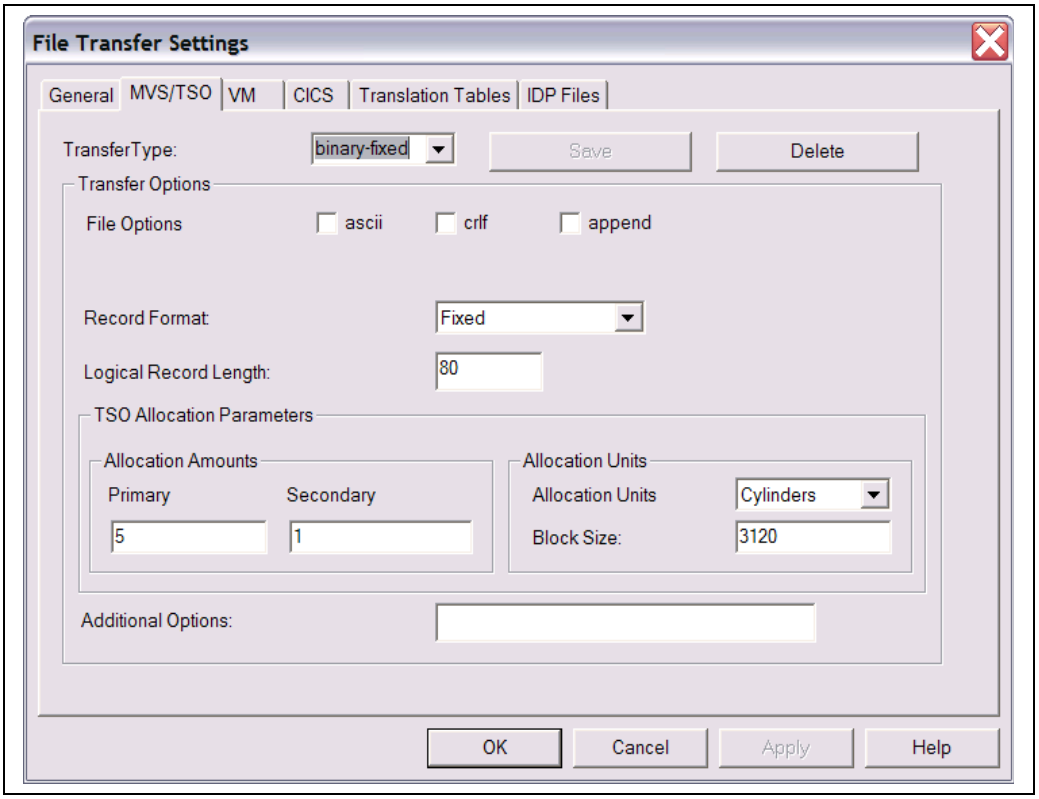
The samples file contains the source for the publish and subscribe programs, the WMQ definitions, the CICS definitions, JCL for the tests, sample data, and an edit exec that can be used to alter the '++' variables in the definitions and JCL. The file must be uploaded to z/OS in binary fixed length record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

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

1. Upload the QSU2_SOURCE_XMIT.bin file, making sure that the target file is defined as a fixed (80 byte length) blocked (3120) file.



2. The binary-fixed transfer type has the following characteristics



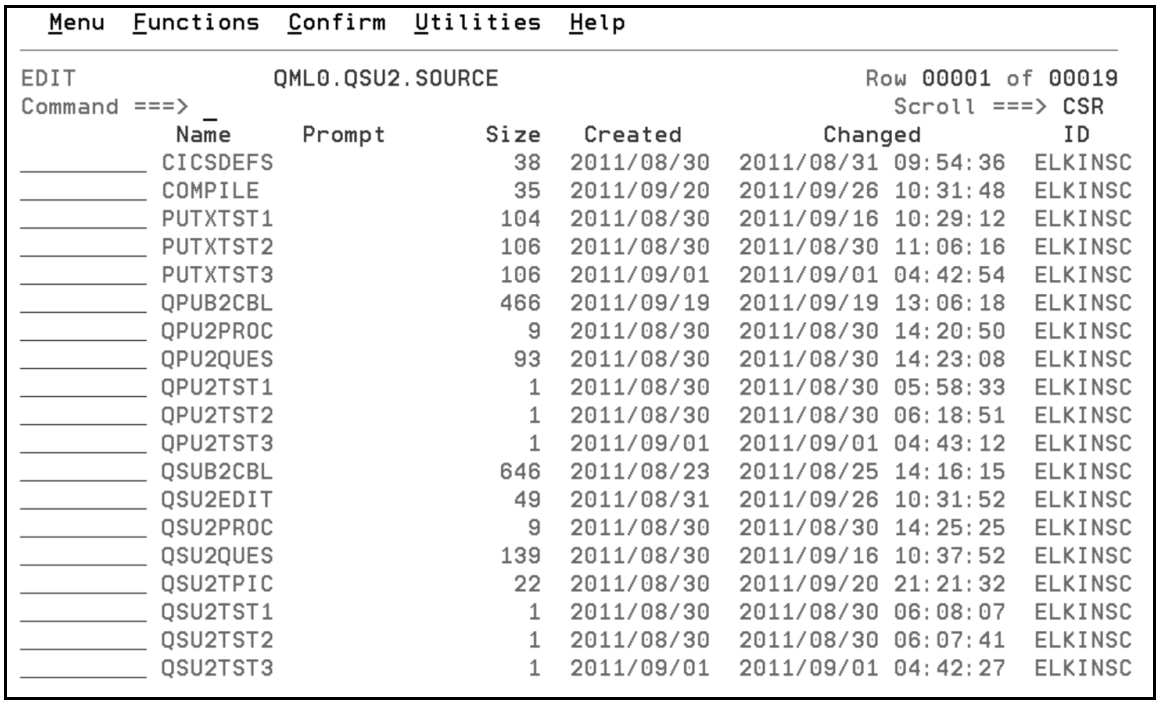
3. Once uploaded, issue the TSO 'RECEIVE command', as shown:

```
 READY
RECEIVE INDSNAME('ELKINSC.QSU2XMIT.BIN')
 INMR901I Dataset QML0.QSU2.SOURCE from ELKINSC on WSC300
 INMR906A Enter restore parameters or 'DELETE' or 'END' +
_
```

4. Respond to the restore parameter request with the dataset name appropriate for your environment.

```
 READY
RECEIVE INDSNAME('ELKINSC.QSU2XMIT.BIN')
 INMR901I Dataset QML0.QSU2.SOURCE from ELKINSC on WSC300
 INMR906A Enter restore parameters or 'DELETE' or 'END' +
DSNAME('ELKINSC.QSU2._SOURCE')
```

5. The PDS created should contain the members as shown.

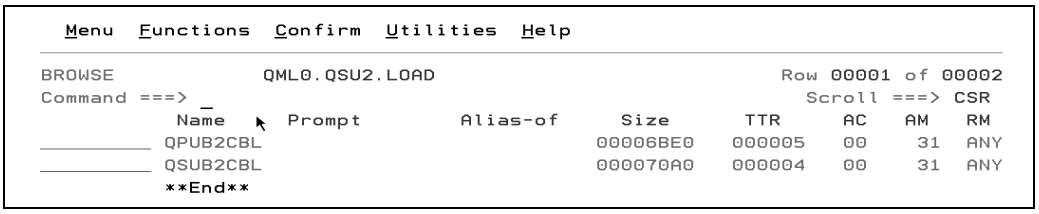


```
  Menu  Functions  Confirm  Utilities  Help
 ------------------------------------------------------------------------------
EDIT              QML0.QSU2.SOURCE                          Row 00001 of 00019
Command ===> _                                               Scroll ===> CSR
           Name     Prompt       Size   Created          Changed           ID
_________ CICSDEFS                 38  2011/08/30  2011/08/31 09:54:36  ELKINSC
_________ COMPILE                  35  2011/09/20  2011/09/26 10:31:48  ELKINSC
_________ PUTXTST1                104  2011/08/30  2011/09/16 10:29:12  ELKINSC
_________ PUTXTST2                106  2011/08/30  2011/08/30 11:06:16  ELKINSC
_________ PUTXTST3                106  2011/09/01  2011/09/01 04:42:54  ELKINSC
_________ QPUB2CBL                466  2011/09/19  2011/09/19 13:06:18  ELKINSC
_________ QPU2PROC                  9  2011/08/30  2011/08/30 14:20:50  ELKINSC
_________ QPU2QUES                 93  2011/08/30  2011/08/30 14:23:08  ELKINSC
_________ QPU2TST1                  1  2011/08/30  2011/08/30 05:58:33  ELKINSC
_________ QPU2TST2                  1  2011/08/30  2011/08/30 06:18:51  ELKINSC
_________ QPU2TST3                  1  2011/09/01  2011/09/01 04:43:12  ELKINSC
_________ QSUB2CBL                646  2011/08/23  2011/08/25 14:16:15  ELKINSC
_________ QSU2EDIT                 49  2011/08/31  2011/09/26 10:31:52  ELKINSC
_________ QSU2PROC                  9  2011/08/30  2011/08/30 14:25:25  ELKINSC
_________ QSU2QUES                139  2011/08/30  2011/09/16 10:37:52  ELKINSC
_________ QSU2TPIC                 22  2011/08/30  2011/09/20 21:21:32  ELKINSC
_________ QSU2TST1                  1  2011/08/30  2011/08/30 06:08:07  ELKINSC
_________ QSU2TST2                  1  2011/08/30  2011/08/30 06:07:41  ELKINSC
_________ QSU2TST3                  1  2011/09/01  2011/09/01 04:42:27  ELKINSC
```

Uploading the loadlib file

The samples file contains the COBOL source that can be compiled and linked into a load library included in the CICS RPL list. Alternatively, the load modules are included in the QSU2_LOAD_XMIT.bin file included with this TechDoc. The upload steps for the load library file are the same as those for the source. They are:

1. Upload the QSU2_LOAD_XMIT.bin file, making sure that the target file is defined as a fixed (80 byte length) blocked (3120) file.
2. Once uploaded, issue the TSO 'RECEIVE command'.
3. Respond to the restore parameter request with the dataset name appropriate for your environment.
4. The PDS created should contain the members as shown.



5. The load library can be included in the CICS RPL, or the load modules can be copied to an existing library.

Edit and use the REXX Exec

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 QSU2EDIT member in edit mode.

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       QML0.QSU2.SOURCE(QSU2EDIT) - 01.02                Columns 00001 00080
Command ===> ■                                                  Scroll ===> CSR
****** ********************************* Top of Data ***********************************
000001 ISREDIT MACRO NOPROCESS
000002 ADDRESS ISREDIT
000003 /* **** CICS DEFINITIONS CHANGES ****  */
000004 "CHANGE '++QML0GRP++' 'CICSGRP' ALL"
000005 "change '++QPU2++' 'QPU2' all"
000006 "change '++QSU2++' 'QSU2' all"
000007 /* **** OEMPUTX CHANGES ****  */
000008 "change '++IP13.LOADLIB++' 'SYS1.MQM.IP13.LOADLIB' all"
000009 "change '++WMQHLQ++' 'SYS1.MQV701' all"
000010 "change '++THIS.PDSNAME++' 'QSU2.INSTALL.PDS' all"
000011 "change '++QMGR++' 'CSQ1' all"
000012 "change '++DOC.SUMMARY++' 'ELKINSC.IP13.DOC.SUMARY' all"
000013 "change '++DB2.NAME++' 'DSNA' all"
000014 "change '++DOC.SUMMARY++' 'ELKINSC.IP13.DOC.SUMARY' all"
000015 "change '++IP13.TEMP++' 'ELKINSC.IP13.TEMP' all"
000016 "change '++QSU2.CONTROL.QUEUE++'  'QSU2.CONTROL.QUEUE' all"
000017 "change '++QPU2.CONTROL.QUEUE++'  'QPU2.CONTROL.QUEUE' all"
000018 "change '++QSU2.STATUS.QUEUE++'  'QSU2.STATUS.QUEUE' all"
000019 "change '++QPU2.STATUS.QUEUE++'  'QPU2.STATUS.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 'CICSGRP' was changed to 'MQTSTGRP' because that was the name of the RDO group being used for WMQ sample programs.

   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('QML0.QSU2.SOURCE')

4) Apply the edits by entering the 'QSU2EDIT' command to alter the ++ variables to those valid in your environment.

```
EDIT       ELKINSC.QSU2.SOURCE(CICSDEFS) - 01.00          Columns 00001 00072
Command ===> qsu2edit                                       Scroll ===> CSR
****** ***************************** Top of Data ******************************
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>           data does not contain any lower case characters.
000001  DEFINE PROGRAM(QPUB2CBL) GROUP(++QML0GRP++)
000002  DESCRIPTION(SAMPLE WMQ PUBLICATION PROGRAM)
000003         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000004         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000005         EXECKEY(USER) CONCURRENCY(QUASIRENT) 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(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
000009  DEFINE PROGRAM(QSUB2CBL) GROUP(++QML0GRP++)
000010  DESCRIPTION(SAMPLE WMQ SUBSCRIPTION PROGRAM)
000011         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000012         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000013         EXECKEY(USER) CONCURRENCY(QUASIRENT) API(CICSAPI) DYNAMIC(NO)
000014         EXECUTIONSET(FULLAPI) JVM(NO) JVMPROFILE(DFHJVMPR)
000015         DEFINETIME(11/08/24 19:47:51) CHANGETIME(11/08/24 19:48:09)
000016         CHANGEUSRID(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
```

5) The ++ values should be altered to those entered in the REXX exec, as shown:

```
EDIT       ELKINSC.QSU2.SOURCE(CICSDEFS) - 01.01          Columns 00001 00072
Command ===>                                                Scroll ===> CSR
****** ***************************** Top of Data ******************************
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>           data does not contain any lower case characters.
==CHG>  DEFINE PROGRAM(QPUB2CBL) GROUP(MQTSTGRP)
000002  DESCRIPTION(SAMPLE WMQ PUBLICATION PROGRAM)
000003         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000004         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000005         EXECKEY(USER) CONCURRENCY(QUASIRENT) 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(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
==CHG>  DEFINE PROGRAM(QSUB2CBL) GROUP(MQTSTGRP)
000010  DESCRIPTION(SAMPLE WMQ SUBSCRIPTION PROGRAM)
000011         LANGUAGE(COBOL) RELOAD(NO) RESIDENT(NO) USAGE(NORMAL)
000012         USELPACOPY(NO) STATUS(ENABLED) CEDF(YES) DATALOCATION(ANY)
000013         EXECKEY(USER) CONCURRENCY(QUASIRENT) API(CICSAPI) DYNAMIC(NO)
000014         EXECUTIONSET(FULLAPI) JVM(NO) JVMPROFILE(DFHJVMPR)
000015         DEFINETIME(11/08/24 19:47:51) CHANGETIME(11/08/24 19:48:09)
000016         CHANGEUSRID(ELKINSC) CHANGEAGENT(CSDAPI) CHANGEAGREL(0660)
==CHG>  DEFINE TRANSACTION(QPU2) GROUP(MQTSTGRP)
000018  DESCRIPTION(SAMPLE WMQ PUBLICATION TRANSACTION)
```

6) Continue using the REXX exec against the other member in the PDS, except for the Exec itself.

CICS Definitions

There are four CICS definitions required; two transactions (QSU2 and QPU2) and the two COBOL programs, QSU2CBL and QPU2CBL 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

Appendix A on page 28 walks thru an example of using RDO to define one of the transactions and COBOL program, if that definition technique is preferred.

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 members QPU2PROC and QSU2PROC are the process definitions used to trigger the QPU2 and QSU2 transactions. 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 to the MQ Explorer.

Queue Definitions

The members QPU2QUES and QSU2QUES are the queue definitions required to implement these samples. 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 to the MQ Explorer.

Topic Definition

The member QSU2TPIC is the topic definition required to implement these samples. The PDS members may be used as input to he CSQUTIL program to define the topic in batch mode, or the object can be defined online via MQ Explorer.

Testing the Sample Programs

Three tests are provided as sample, in the source library they are called PUTXTST1, PUTXTST2, and PUTXTST3. They execute the OEMPUTX program to start both the subscription and publication transactions.

Testing Results – PUTXTST1

PUTXTST1 uses an equal number of publications and subscriptions, in the delivered sample they are both 2. Publications and the subscription are made to the combination of the QSU2.TOPIC and the supplied topic string "TEST/LEVEL 1 ". Note that there is a space following the "LEVEL 1". The delimiter for the topic string is an open parenthesis ('(').

The results are extracted from the OEMPUTX output. Similar results should be seen in other environments.

Publication Control Message:

```
** MESSAGE DATA (80 of 80 bytes printed) **
 00000000 : F26BD8E2 E4F24BE3 D6D7C9C3 6B4D6BE3  2,QSU2.TOPIC,(,T
 00000010 : C5E2E361 D3C5E5C5 D340F140 4D404040  EST/LEVEL 1 (
 00000020 : 40404040 40404040 40404040 40404040
 00000030 : 40404040 40404040 40404040 40404040
 00000040 : 40404040 40404040 40404040 40404040
```

Subscription Control Message:

```
** MESSAGE DATA (80 of 80 bytes printed) **
 00000000 : F26BD8E2 E4F24BE2 E4C24BD8 E4C5E4C5  2,QSU2.SUB.QUEUE
 00000010 : 6BD8E2E4 F24BE3D6 D7C9C36B 4D6BE3C5  ,QSU2.TOPIC,(,TE
 00000020 : E2E361D3 C5E5C5D3 40F1404D 6B406BD8  ST/LEVEL 1 (, ,Q
 00000030 : E2E4F260 E3C5E2E3 F1404040 40404040  SU2-TEST1
 00000040 : 40404040 40404040 40404040 40404040
```

Publication Status Message:

```
** MESSAGE DATA (513 of 513 bytes printed) **
 00000000 : D7E4C2D3 C9C3C1E3 C9D6D5E2 40D4C1C4  PUBLICATIONS MAD
 00000010 : C57E4040 F0F0F0F0 F240C6D6 D940E3D6  E=  00002 FOR TO
 00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
 00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
 00000040 : 40404040 40404040 40404040 40404040
 00000050 : 40404040 40404040 40404040 4040D9C5                RE
```

```
 00000060 : E2D6D3E5 C5C44040 7E404040 40404040  SOLVED  =
 00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
 00000080 : 40F14040 40404040 40404040 40404040   1
```

Subscription Status Message:

```
** MESSAGE DATA (513 of 513 bytes printed) **
 00000000 : E2E4C2E2 C3D9C9D7 E3C9D6D5 40D4E2C7  SUBSCRIPTION MSG
 00000010 : E27E4040 F0F0F0F0 F240C6D6 D940E3D6  S=  00002 FOR TO
 00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
 00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
 00000040 : 40404040 40404040 40404040 40404040
 00000050 : 40404040 40404040 40404040 4040C6D6                FO
 00000060 : D940D9C5 E2D6D3E5 C5C4407E 40404040  R RESOLVED =
 00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
 00000080 : 40F14040 40404040 40404040 40404040   1
```

Testing Results – PUTXTST2

PUTXTST2 uses an unequal number of publications and subscriptions, there are 900 messages published and 50 messages retrieved from the subscription queue. Publications and the subscription are made to the combination of the QSU2.TOPIC and the supplied topic string "TEST/LEVEL 1 ". Note that there is a space following the "LEVEL 1". The delimiter for the topic string is low values – x'00'.

One note, following this test, there may be 'extra' messages on the subscription destination queue left after the test. This is expected because the subscription remained active until the subscription transaction ended; publications continue to be delivered to the queue after the subscribing application had closed the queue.

The results are extracted from the OEMPUTX output. Similar results should be seen when testing in your environments.

Publication Control Message:

```
** MESSAGE DATA (80 of 80 bytes printed) **
00000000 : F9F0F06B D8E2E4F2 4BE3D6D7 C9C36B00  900,QSU2.TOPIC,.
00000010 : 6BE3C5E2 E361D3C5 E5C5D340 F1004040  ,TEST/LEVEL 1.
00000020 : 40404040 40404040 40400040 40404040           .
00000030 : 40404040 40404040 40404040 40404040
00000040 : 40404040 40404040 40404040 40404040
```

Subscription Control Message:

```
** MESSAGE DATA (80 of 80 bytes printed) **
00000000 : F5F06BD8 E2E4F24B E2E4C24B D8E4C5E4  50,QSU2.SUB.QUEU
00000010 : C56BD8E2 E4F24BE3 D6D7C9C3 6B006BE3  E,QSU2.TOPIC,.,T
00000020 : C5E2E361 D3C5E5C5 D340F100 6B406BD8  EST/LEVEL 1., ,Q
00000030 : E2E4F260 E3C5E2E3 F1404040 40404040  SU2-TEST1
00000040 : 40404040 40404040 40404040 40404040
```

Publication Status Message:

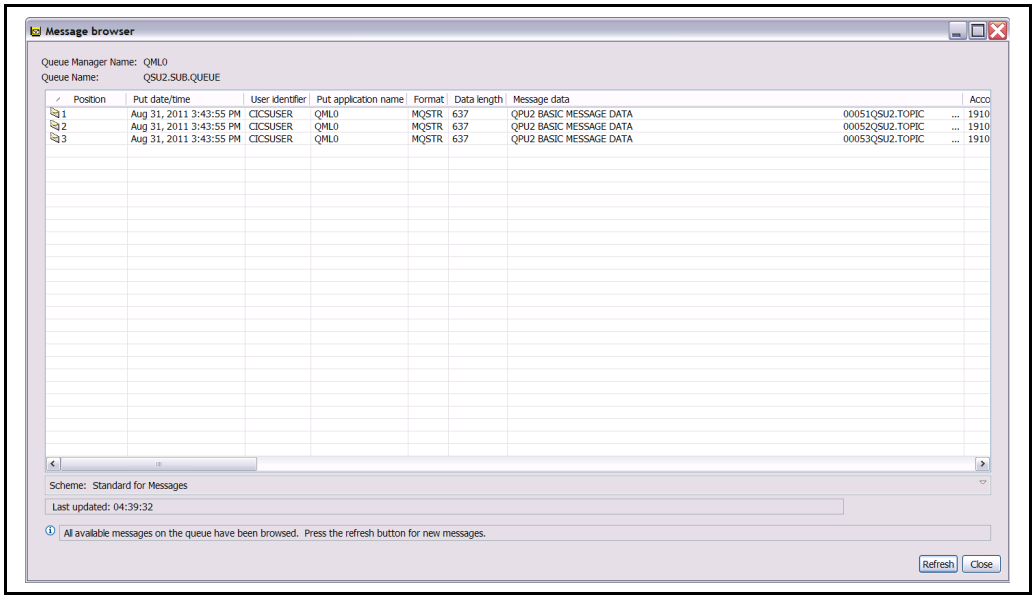
```
** MESSAGE DATA (513 of 513 bytes printed) **
 00000000 : D7E4C2D3 C9C3C1E3 C9D6D5E2 40D4C1C4  PUBLICATIONS MAD
 00000010 : C57E4040 F0F0F9F0 F040C6D6 D940E3D6  E=  00900 FOR TO
 00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
 00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
 00000040 : 40404040 40404040 40404040 40404040
 00000050 : 40404040 40404040 40404040 4040D9C5                RE
```

```
 00000060 : E2D6D3E5 C5C44040 7E404040 40404040  SOLVED  =
 00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
 00000080 : 40F14040 40404040 40404040 40404040   1
```

Subscription Status Message:

```
** MESSAGE DATA (513 of 513 bytes printed) **
 00000000 : E2E4C2E2 C3D9C9D7 E3C9D6D5 40D4E2C7  SUBSCRIPTION MSG
 00000010 : E27E4040 F0F0F0F5 F040C6D6 D940E3D6  S=  00050 FOR TO
 00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
 00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
 00000040 : 40404040 40404040 40404040 40404040
 00000050 : 40404040 40404040 40404040 4040C6D6                FO
 00000060 : D940D9C5 E2D6D3E5 C5C4407E 40404040  R RESOLVED =
 00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
 00000080 : 40F14040 40404040 40404040 40404040   1
```

Remaining Messages on Subscription Destination Queue:



Testing Results – PUTXTST3

PUTXTST3 uses an unequal number of publications and subscriptions, in the delivered sample there are 900 publications and only 50 subscriptions messages will be retrieved. Publications are made to the combination of the QSU2.TOPIC and the supplied topic string "TEST/LEVEL 2 ". Subscriptions are made to the combination of the QSU2.TOPIC and the supplied topic string "TEST/LEVEL 1 ". The delimiter for the topic string is low values. **There should not be any subscription messages retrieved because the resolved topic strings do not match.**

The results are extracted from the OEMPUTX output. Similar results should be seen in other environments. Also note that due to the get wait on the subscription transaction, that job will not finish until well after the publication job has completed.

Please note that if the subscription destination queue is not cleared prior to running this test, the remaining subscription messages from the previous test will be picked up. This is expected behavior, as there is nothing identifying those messages as 'belonging' to the previous subscription. If a unique correlation ID had been used on each of the subscription requests, no messages will be returned on the MQGET.

Publication Control Message:

```
** MESSAGE DATA (80 of 80 bytes printed) **
00000000 : F9F0F06B D8E2E4F2 4BE3D6D7 C9C36B00  900,QSU2.TOPIC,.
00000010 : 6BE3C5E2 E361D3C5 E5C5D340 F2004040  ,TEST/LEVEL 2.
00000020 : 40404040 40404040 40400040 40404040            .
00000030 : 40404040 40404040 40404040 40404040
00000040 : 40404040 40404040 40404040 40404040
```

Subscription Control Message:

```
 ** MESSAGE DATA (80 of 80 bytes printed) **
00000000 : F5F06BD8 E2E4F24B E2E4C24B D8E4C5E4  50,QSU2.SUB.QUEU
00000010 : C56BD8E2 E4F24BE3 D6D7C9C3 6B006BE3  E,QSU2.TOPIC,.,T
00000020 : C5E2E361 D3C5E5C5 D340F100 6B406BD8  EST/LEVEL 1., ,Q
00000030 : E2E4F260 E3C5E2E3 F1404040 40404040  SU2-TEST1
00000040 : 40404040 40404040 40404040 40404040
```

Publication Status Message:

```
** MESSAGE DATA (513 of 513 bytes printed) **
00000000 : D7E4C2D3 C9C3C1E3 C9D6D5E2 40D4C1C4  PUBLICATIONS MAD
00000010 : C57E4040 F0F0F9F0 F040C6D6 D940E3D6  E=  00900 FOR TO
00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
```

```
00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
00000040 : 40404040 40404040 40404040 40404040
00000050 : 40404040 40404040 40404040 4040D9C5                RE
00000060 : E2D6D3E5 C5C44040 7E404040 40404040  SOLVED  =
00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
00000080 : 40F24040 40404040 40404040 40404040   2
```

Subscription Status Message:

```
** MESSAGE DATA (513 of 513 bytes printed) **
00000000 : E2E4C2E2 C3D9C9D7 E3C9D6D5 40D4E2C7  SUBSCRIPTION MSG
00000010 : E27E4040 F0F0F0F0 F040C6D6 D940E3D6  S=  00000 FOR TO
00000020 : D7C9C340 7E404040 40404040 40D8E2E4  PIC =        QSU
00000030 : F24BE3D6 D7C9C340 40404040 40404040  2.TOPIC
00000040 : 40404040 40404040 40404040 40404040
00000050 : 40404040 40404040 40404040 4040C6D6                FO
00000060 : D940D9C5 E2D6D3E5 C5C4407E 40404040  R RESOLVED =
00000070 : 40D8E2E4 F261E3C5 E2E361D3 C5E5C5D3   QSU2/TEST/LEVEL
00000080 : 40F14040 40404040 40404040 40404040   1
```

Appendix A – CICS RDO definition sample

This information is provided for the WMQ administrators who may not be familiar with defining resources in CICS.

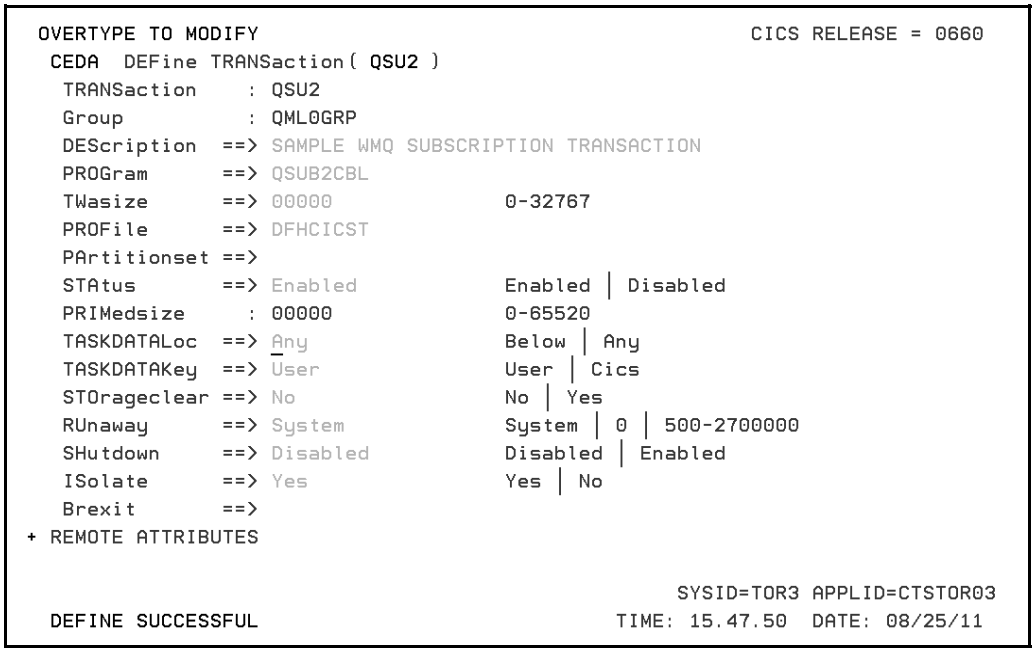
QSU2 definition:

Using CEDA the QSU2 definition was created using the following steps:

1) Enter the CEDA defining the transaction and the associated program to a specified group; in the example the group QML0GRP is used.

```
ceda def trans(qsu2) group(qml0grp)
```

2) The definition success message should appear as shown. Note that the TASKDATALoc can be ‘Any’.



QSUB2CBL Definition:

1) Enter the define program command, supplying the group name.

```
ceda def prog(qsub2cbl) group(qml0grp)
```

2) Set the language to COBOL, the data location to any and hit the enter key. The 'Define Successful' message should be displayed as follows.

```
 DEF prog(QSUB2CBL) GROUP(QML0GRP)
 OVERTYPE TO MODIFY                                      CICS RELEASE = 0660
  CEDA  DEFine PROGram( QSUB2CBL )
   PROGram        : QSUB2CBL
   Group          : QML0GRP
   DEScription  ==> Sample COBOL WMQ subscription program
   Language     ==> CO                COBol | Assembler | Le370 | C | Pli
   RELoad       ==> No                No | Yes
   RESident     ==> No                No | Yes
   USAge        ==> Normal            Normal | Transient
   USElpacopy   ==> No                No | Yes
   Status       ==> Enabled           Enabled | Disabled
   RSl            : 00                0-24 | Public
   CEdf         ==> Yes               Yes | No
   DAtalocation ==> Any               Below | Any
   EXECKey      ==> User              User | Cics
   COncurrency  ==> Quasirent         Quasirent | Threadsafe
   Api          ==> Cicsapi           Cicsapi | Openapi
  REMOTE ATTRIBUTES
+  DYnamic      ==> No                No | Yes

                                                  SYSID=TOR3 APPLID=CTSTOR03
  DEFINE SUCCESSFUL                             TIME: 15.44.11  DATE: 08/25/11
PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Please note that the programs may be defined with a concurrency attribute of 'Threadsafe'.

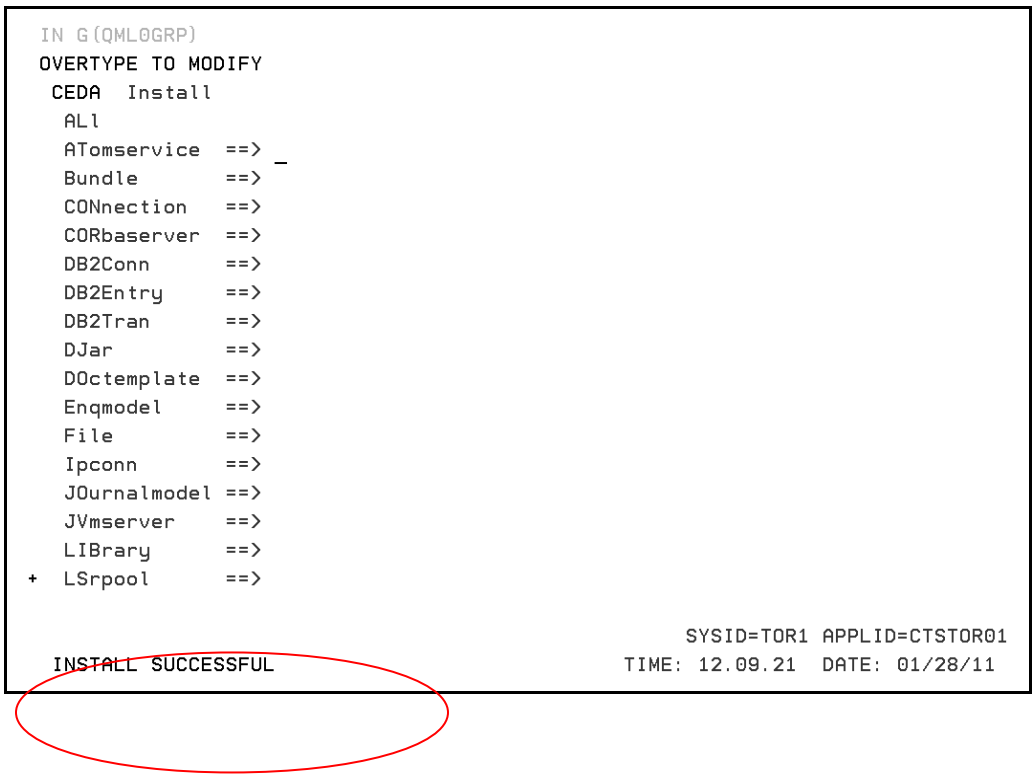
QML0GRP Installation

The new resources must be installed into the region.

1) Enter the CEDA install command as shown:


```
CEDA IN G(QML0GRP)
```

2) The install successful messages should be displayed as shown:



Appendix B – Sample Compile and Link JCL

This sample JCL compiles and links the COBOL programs from this sample. Note that the 'MQ Stub' that is lined with the COBOL program is the CICS supplied stub, DFHMQSTB.

```
//ADD JOBCARD HERE                                                      00020001
//*                                                                     00030000
//ADDPROCS JCLLIB ORDER=(++COBOL.LIB++.SIGYPROC)                        00040001
//*                                                                     00050000
//* TRANSLATE, COMPILE, AND LINK A COBOL CICS PROGRAM                   00060000
//*                                                                     00070000
//COMPCB   EXEC PROC=IGYWCL,                                            00080000
//         LNGPRFX='++COBOL.LIB++',                                     00090001
//         LIBPRFX='++LE.LIB++',                                        00100001
//         PGMLIB='++YOUR.LOADLIB++',                                   00110001
//         GOPGM='QPUB2CBL',                                            00120000
//         PARM.COBOL='CICS,LIB,NODYNAM,RENT,SIZE(4000K),LIST,MAP'      00130000
//STEPLIB  DD DSN=++COBOL.LIB++.SIGYCOMP,DISP=SHR                       00140001
//             DD DISP=SHR,DSN=++CICS.LIB++.SDFHLOAD                    00150001
//* PROGRAM TO COMPILE GOES IN NEXT STATEMENT                          00160000
//COBOL.SYSIN  DD DISP=SHR,DSN=++SOURCE.PDSNAME++(QPUB2CBL)             00170101
//COBOL.SYSLIN DD DISP=SHR,DSN=++OBJECT.PDSNAME++(&GOPGM)               00170201
//COBOL.SYSLIB DD DISP=SHR,DSN=++WMQHLQ++.SCSQCOBC                      00171101
//             DD DISP=SHR,DSN=++SOURCE.PDSNAME++                       00172001
//             DD DISP=SHR,DSN=++CICS.LIB++.ADFHCOB                     00173001
//LKED.SYSLIB  DD DISP=SHR,DSN=++CICS.LIB++.SDFHLOAD                    00190001
//             DD DSN=++LE.LIB++.SCEELKED,DISP=SHR                      00200001
//             DD DISP=SHR,DSN=++WMQHLQ++UPD.SCSQAUTH                   00201001
//             DD DISP=SHR,DSN=++WMQHLQ++UPD.SCSQLOAD                   00203001
//LKED.OBJECT  DD DISP=SHR,DSN=++OBJECT.PDSNAME++                       00204001
//LKED.SYSLIN DD DISP=SHR,DSN=++OBJECT.PDSNAME++(&GOPGM)                00205001
//LKED.SYSIN   DD *                                                     00206000
        INCLUDE SYSLIB(DFHMQSTB)                                        00207000
        INCLUDE OBJECT(QPUB2CBL)                                        00208000
        ENTRY QPUB2CBL                                                  00209000
        NAME QPUB2CBL(R)                                                00209100
/*                                                                      00209200
//*                                                                     00209300
//LKED.SYSLMOD DD DISP=SHR,DSNAME=&PGMLIB(&GOPGM)                       00211000
```


Acknowledgments:

The authors would like to thank the following people for their assistance
Mark Taylor
Shalawn King
Jenifer Foley
Chris Griego
Ashley Golden