Messaging and Queuing Extensions for VSE/ESA

Document Number GG24-4296-00

May 1994

International Technical Support Organization
Boeblingen Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xi.

First Edition (May 1994)

This edition applies to ezBRIDGE** Transact** on VSE/ESA* for the MQSeries*, program number 5758-SM1 (in the United States of America) and 5787-ECX (outside the USA) for use with VSE/ESA.

Order publications through your IBM* representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 3222 Building 71032-02
Postfach 1380
71032 Boeblingen, Germany

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document describes ezBRIDGE Transact on VSE/ESA for MQSeries. It provides guidance on the implementation of a batch interface to the product, which, as shipped, provides a programming interface to CICS/VSE* application programs written in VS COBOL II.

The document also gives some sample definitions which will simplify the planning and implementation process for ezBRIDGE Transact on VSE/ESA.

This document was written for IBM Technical Professionals, and for customer technical personnel involved in the planning implementation and administration of ezBRIDGE Transact on VSE/ESA for MQSeries systems.

Knowledge of the principles of the messaging and queuing interface used by MQSeries is assumed at least to the Concepts and Architecture level.

MR (121 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IBM Technical Professionals and Customer Technical and Operations Staff to extend the capabilities of ezBRIDGE Transact on VSE/ESA for MQSeries to allow it to be accessed from VSE/ESA batch applications. It is also designed to help in verifying remote connections to other ezBRIDGE Transact on VSE/ESA for MQSeries systems or to other members of the MQSeries. The information in this publication is not intended as the specification of any programming interfaces that are provided by ezBRIDGE Transact on VSE/ESA for MQSeries. See the PUBLICATIONS section of the IBM Programming Announcement for ezBRIDGE Transact on VSE/ESA for MQSeries for more information about what publications are considered to be product documentation.

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 product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this 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 may have 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.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

AIX
AIX/6000
BookManager
CICS
CICS/ESA
CICS/VSE
IBM
MQSeries
MVS/ESA*
OS/2
OS/400
PS/2
VM/ESA
VSE/ESA
VTAM

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

ezBRIDGE
System Strategies Inc. (Reg. Trademark)
Transact
System Strategies Inc.

Preface

This publication is intended to help IBM Technical Professionals and Customer Technical and Operations Staff to extend the capabilities of ezBRIDGE Transact on VSE/ESA for MQSeries to allow it to be accessed from VSE/ESA batch applications. It is also designed to help in verifying remote connections to other ezBRIDGE Transact on VSE/ESA for MQSeries systems or to other members of the MQSeries.

This document contains sample definitions which will allow implementation of a simple distributed network, which can be used for verification of both local and remote queues. It also contains sample programs which can be used to extend the ezBRIDGE Transact on VSE/ESA for MQSeries capabilities to batch applications running in VSE/ESA.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “What This Book Describes”

  This provides a description of the motivation for this book, and how additional objectives were added during its development.

- Chapter 2, “Some MQI Principles Applied to the ITSC Example”

  This provides a brief review of some MQSeries definitions and principles, and simplifies them by relating them to the ITSO Boeblingen′s sample implementation.

- Chapter 3, “Planning and Installation”

  This provides guidance on planning and installing ezBRIDGE Transact on VSE/ESA for MQSeries, and provides instructions for verification of distributed queuing.

- Chapter 4, “A Sample Batch Interface for ezBRIDGE Transact for VSE/ESA”

  This describes the design and implementation of the sample batch interface for ezBRIDGE Transact on VSE/ESA for MQSeries.

- Chapter 5, “Differences Between ezBRIDGE Transact on VSE/ESA for MQSeries and Other MQSeries Implementations”

  This lists some of the items to consider when implementing MQSeries products on multiple platforms, or when implementing ezBRIDGE Transact on VSE/ESA for MQSeries, where other platforms may later be added to the network.

- Appendix A, “Planning Worksheets Used in the Sample”

  This appendix provides worksheets used in the implementation of the ITSO sample system. Additional notes are provided to guide in completing the worksheets, and in relating the worksheets to each other.

- Appendix B, “Jobs Used to Install ezBRIDGE Transact on VSE/ESA”

  This appendix documents the job-streams used to install ezBRIDGE Transact on VSE/ESA for MQSeries.

- Appendix C, “APPC Definitions Used to Communicate Between PRODCICS and CICSF8”

  This appendix lists the CICS* and VTAM* definitions used to establish distributed queuing between the two CICS/VSE systems.

- Appendix D, “Definitions Used for Distributed Queue Processing”

  This appendix lists the ezBRIDGE definitions used to establish distributed queuing between the two CICS/VSE systems. It also provides samples of the output written to the CICS/VSE Message Log when transmission to a remote system is successfully initiated.

- Appendix E, “Components for Compiling and Running the Interface Sample”

  This appendix provides listings of the sample programs used to extend the capabilities of ezBRIDGE Transact on VSE/ESA for MQSeries to batch applications.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *An Introduction To Messaging and Queuing*, GC33-0805
- *Messaging And Queuing Series Technical Reference*, SC33-0850
- *ezBRIDGE Transact for MQSERIES Overview Concepts & Architecture*, GC33-1141
- *ezBRIDGE Transact for VSE/ESA for MQSERIES User Manual*, SC33-1142
- *MQSERIES Distributed Queue Management Guide*, SC33-1139
- *CICS/VSE Intercommunication Guide*, SC33-0701
- *CICS Application Programming Primer*, SC33-0674
- *CICS/VSE Application Programming Guide*, SC33-0712
- *CICS/VSE Application Programming Reference*, SC33-0713
- *VSE/ESA System Macro Reference*, SC33-6516
- *VSE/ESA System Macros User's Guide*, SC33-6515
- *VSE/AF V5 Diagnostic Reference Supervisor*, LY33-9127

International Technical Support Organization Publications

- *Selected Examples of Distributed Application Processing Using MQSERIES*, GG24-4167

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of redbooks online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Bulletins (Redbooks)

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

Customers may order hardcopy redbooks individually or in customized sets, called GBOFs, which relate to specific functions of interest. You may also order redbooks in online format on CDROM collections, which contain the redbooks for multiple products.

Acknowledgments

The advisor for this project was:

Peter Lockwood
International Technical Support Organization, Boeblingen Center

The authors of this document are:

Richard Meyer
ISM South Africa

Isamu Miyamoto Okuta
IBM Brasil

This publication is the result of a residency conducted at the International Technical Support Organization, Boeblingen Center.

Thanks to the following person for the invaluable advice and guidance provided in the production of this document:

Ian Craggs
IBM Hursley Laboratories

Chapter 1. What This Book Describes

This book was conceived with the very specific objective of devising and documenting a sample interface which would extend the MQI provided by ezBRIDGE Transact on VSE/ESA for MQSeries to give VSE/ESA batch programs access to the power and flexibility of the MQSeries. During work to meet these objectives it became obvious that some sample definitions would greatly ease the implementation of ezBRIDGE Transact on VSE/ESA for MQSeries for many customers, especially in the area of defining and activating remote channel links.

In addition, it was felt that summarizing some of the current differences between ezBRIDGE Transact on VSE/ESA for MQSeries and other members of the MQSeries family would provide assistance at the planning stage.

Our sample definitions for the remote channel links were limited to those needed to connect two CICS/VSE systems on a single VSE/ESA 1.3.3 host, using LU6.2 connections.

Chapter 2. Some MQI Principles Applied to the ITSC Example

This chapter reviews some MQI principles, as it is felt that the graphic portrayal of systems used here is easier to understand, and more clearly shows the relationship of the different definitions.

2.1 Remote Queues and Transmission Queues

A remote queue definition identifies a queue belonging to another queue manager. This queue must therefore be defined as a local queue to that queue manager.

The information specified for the definition of this remote queue object enables the queue manager to find the remote queue manager, so that any messages destined for a remote queue go to the correct queue manager. Since this means that the routing is achieved by the definition, it increases the flexibility of the application, which merely needs to use the queue name, without even needing to know whether it is local or remote.

The normal method of operating for remote definitions should be to use a remote queue, rather than writing directly to the transmission queue.

2.1.1 Use of the Queue Definitions

For the transfer of a message from one queue manager (local) to another queue manager (remote) the definitions used were as follows:

- On SSIQM (the local queue manager)

  XMIT4 This is a transmission queue and is used to transfer information over a remote channel, to the queue manager on that remote system. Although an application may write directly to a transmission queue, the application itself must then provide explicit routing information, with a resulting loss of flexibility. We strongly recommend the use of a remote queue, which routes to the transmission queues, rather than writing directly to the transmission queue.

  F8R This is a remote queue definition which will use the transmission queue XMIT4 to route messages to the remote queue manager SSIF8, to be placed on the queue F8L which is defined to that remote queue manager as local.

- On SSIF8 (the remote queue manager)

  F8L This queue is local to SSIF8, but is a remote queue to SSIQM, which knows it as F8R, or F8L@SSIF8.

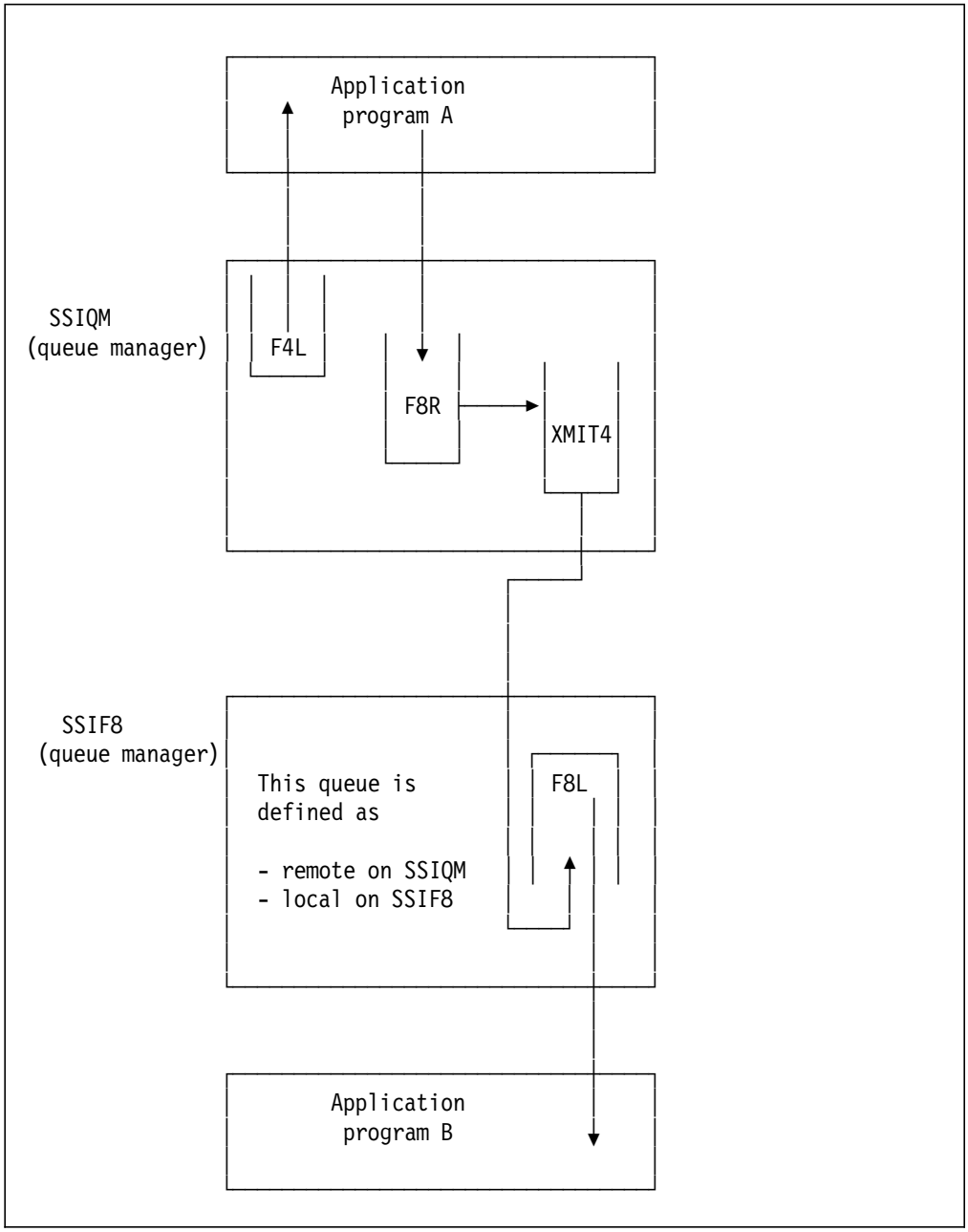


Figure 1. Remote Queue and Transmission Queue. This describes the basic definitions needed for the queues. The values specified are shown in Figure 33 through Figure 38.

Note: Although it is possible to write directly to a transmission queue, the routing information must then be provided from within the application, whereas if a remote destination is used, the application flexibility is enhanced. We used remote queues for our access to remote systems.

2.2 Channels

A channel is a unidirectional point-to-point communications link between two MQseries systems. Messages can flow over that channel in one direction only. The simplest case for communications to a remote system would have a single channel, but there would be no possibility of receiving a reply from the remote system. This should therefore be regarded as a restricted special case, which is shown in Figure 2 on page 6.

For the usual case where MQSeries systems need to exchange information, two channels are required, and this is shown in Figure 3 on page 7.

For outbound channels, ezBRIDGE Transact on VSE/ESA for MQSeries reads messages from the associated transmission queue and sends them to the remote system via the communications link. For inbound channels, ezBRIDGE Transact on VSE/ESA for MQSeries receives messages from the communications link and places them in the destination local queue.

In fact a transaction sending a message to a remote system will probably not write directly to the transmission queue, since if it does it must provide a fully qualified destination address. It is much more sensible to write it to a remote queue which is associated with the transmission queue concerned.

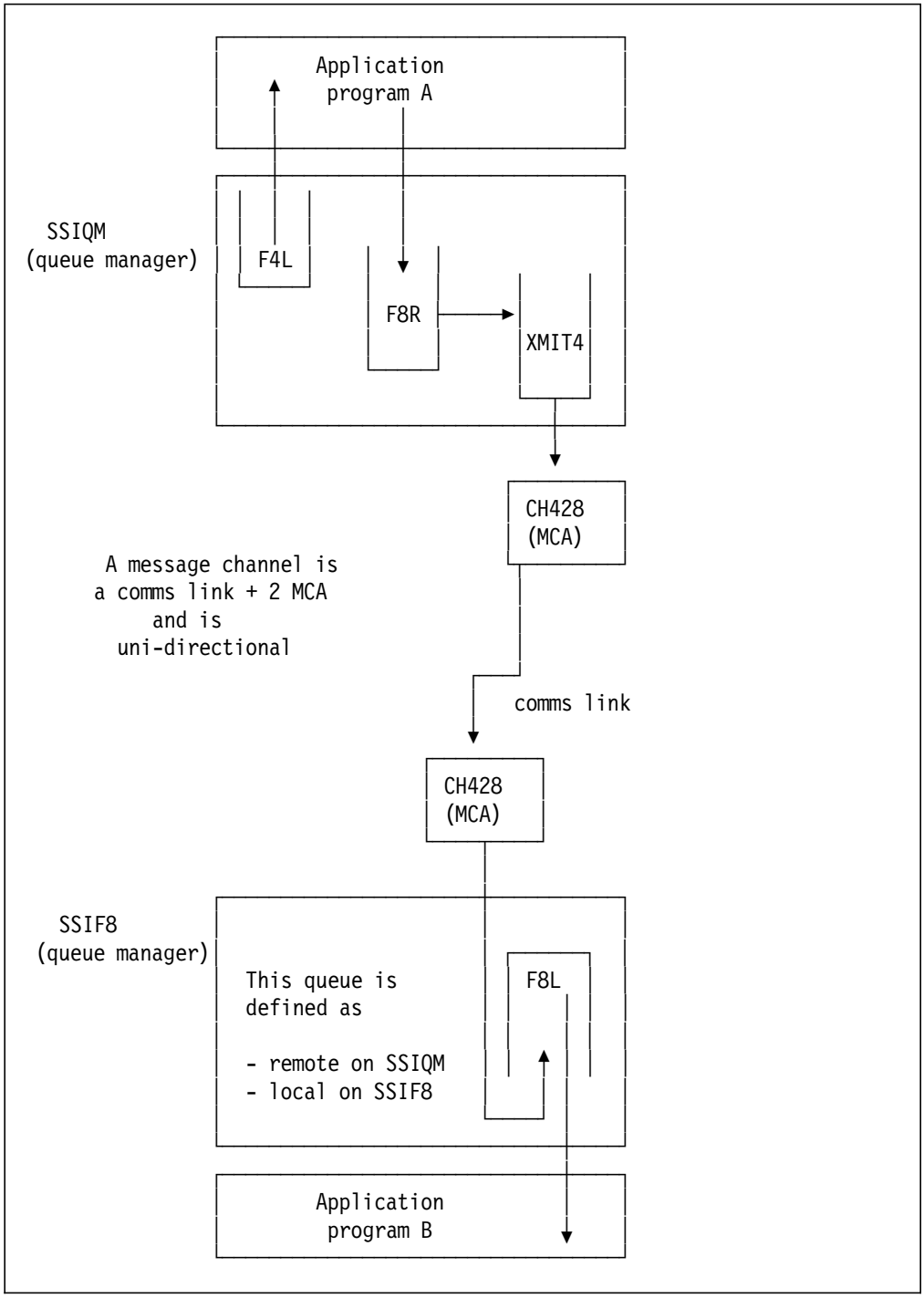


Figure 2. Remote Queue, Transmission Queue and Message Channel. This adds the basic definitions needed for the channels. The values specified are shown in Figure 35 on page 54 and Figure 37 on page 56.

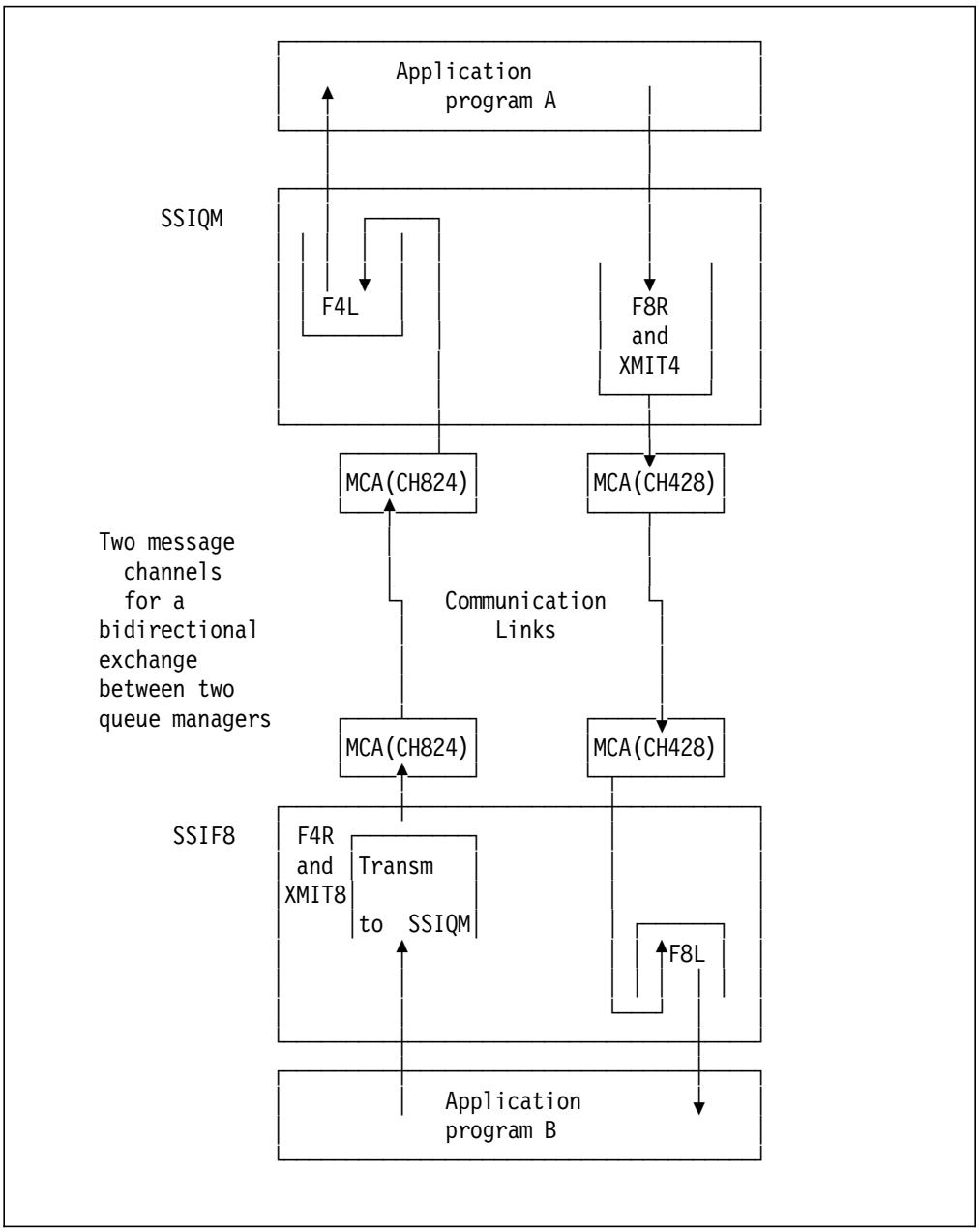


Figure 3. Bi-directional Transmissions. This adds the basic definitions needed for message transmission in both directions. The new definitions are shown in Figure 40 on page 58 through Figure 44 on page 60.

2.3 Triggers and Remote Transmissions

Some applications will run continuously and will always be available to read a message when it arrives on the application's input queue. This will of course use resources, even when there is no message on the queue, because at the least the application must regularly interrogate the queue. This is not desirable when the message traffic is light, or when the volumes fluctuate.

The queue manager provides a triggering facility to ensure that an application can be designed to run only when there are messages to be processed. A local QUEUE definition can have a Trigger event associated with it when it is defined, and the event activates the MQM Trigger API Handler (that is, the MQ02 CICS transaction). Be aware that the MQ02 is triggered implicitly and any definition of it explicitly in the trigger definitions will cause problems.

The trigger API Handler will either issue an EXEC CICS LINK for the application program required, or an EXEC CICS START for the transaction associated with the application. The application will be passed in a DFHCOMMAREA certain data formats which describe the type of event, and the local QUEUE to be read.

The special case we are interested in is the trigger on a transmission queue. For this you should not code a transaction identifier, but should code the program name of MQPSEND which is the remote channel sender program. You can (and should) also specify the remote channel ID which is to be used for the transmission of this queue. As a complement to this you must also specify, in the channel record, what transaction is to be driven on the remote system. For the simple transmission case this should be MQ01, which is the transaction associated with MQPRECV (the channel receiver task).

WARNING

You are allowed, when specifying a trigger, to provide a terminal ID which can be used for debugging purposes. This can then be traced using CEDF. If a terminal name is given, ezBRIDGE Transact on VSE/ESA for MQSeries checks it during initialization. You must therefore ensure that if you wish to trace a triggered task the terminal is predefined, or if it is an auto-installed terminal, that it has already been used (and therefore auto-installed). If this is not the case and you have a terminal associated with a transmission trigger the initialization process suffers a CICS AEIK abend (that is, a TERMIDERR for which no handle condition exists).

This facility is intended for debugging user programs which are triggered, and not programs which form part of ezBRIDGE Transact on VSE/ESA for MQSeries.

The process followed is that the application writes to the remote queue (in our case F8R). This is associated with the transmission queue XMIT4. When information arrives at XMIT4 the channel sender program MQPSEND is triggered for the local end of CH428, and starts the channel agent (MCA) transmitting a request that the channel receiver transaction (MQ01) be run at the remote side to place the data in the queue (defined in SSIQM as remote (F8R) and in SSIF8 as local (F8L).

WARNING

Our testing revealed no method of starting a transmission, apart from by using a trigger in the transmission queue.

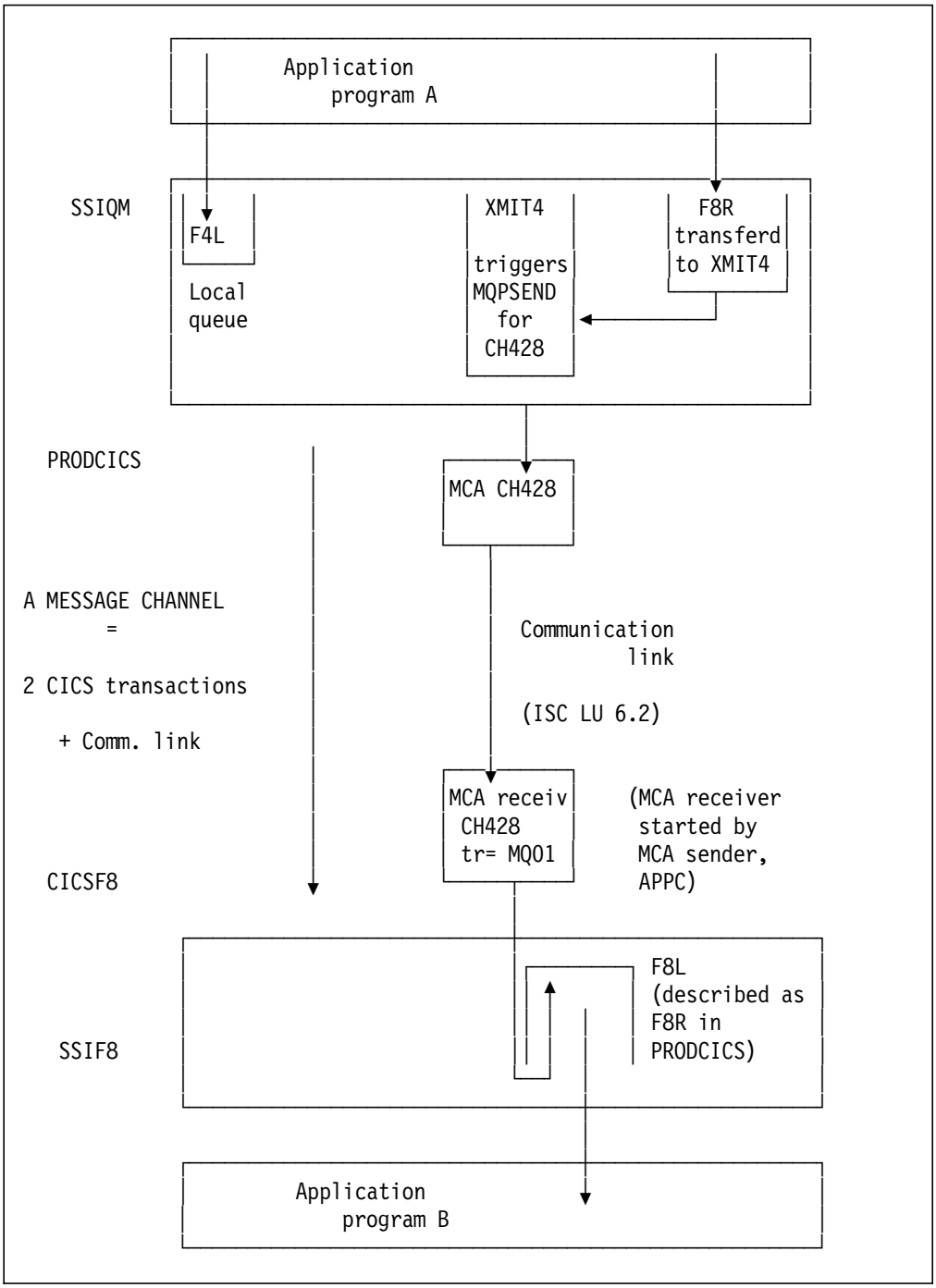


Figure 4. Triggering of Transmissions. This adds the basic definitions needed for the automatic triggering of message transmission. The new definitions are shown in Figure 33 on page 53.

Chapter 3. Planning and Installation

During the installation and enabling of the system to be used for the batch interface development and testing, we encountered a number of situations where hindsight provided a clear indication of where we had gone wrong. These are documented, together with a working set of definitions, so that other users of ezBRIDGE Transact on VSE/ESA for MQSeries can take these definitions, and use them to achieve a smoother and simpler initial installation.

3.1 Planning the Network

As with any new implementation, careful planning is a necessary prerequisite to a good implementation. The ITSO Center Boeblingen implementation was based on using the simplest possible environment which would allow exploitation of access to local queues, but which could still be expanded to allow access to remote queues.

3.1.1 The ITSO System Used

All testing both of the environment as a whole, and also of the batch interface, was done in the same environment. A single VSE/ESA 1.3.3 was used running under VM/ESA Release 2.1 level 9305. All software is listed below:

- VM/ESA 2.1 level 9305
- VSE/ESA 1.3.3

  The environment used was environment 6, that is, MODE=ESA
- VS COBOL II Compiler and Libraries 1.4.0
- VSE/VTAM 3.4
- CICS/VSE 2.2

These software levels exceed the minimum supported levels for ezBRIDGE Transact on VSE/ESA for MQSeries.

The hardware used was as follows:

- 9121 Model 260 with 128MB of Memory
- 3390 model 2 attached by 3990 model 03
- Local non-SNA terminals dialled from VM, using auto-install
- Disks were dedicated

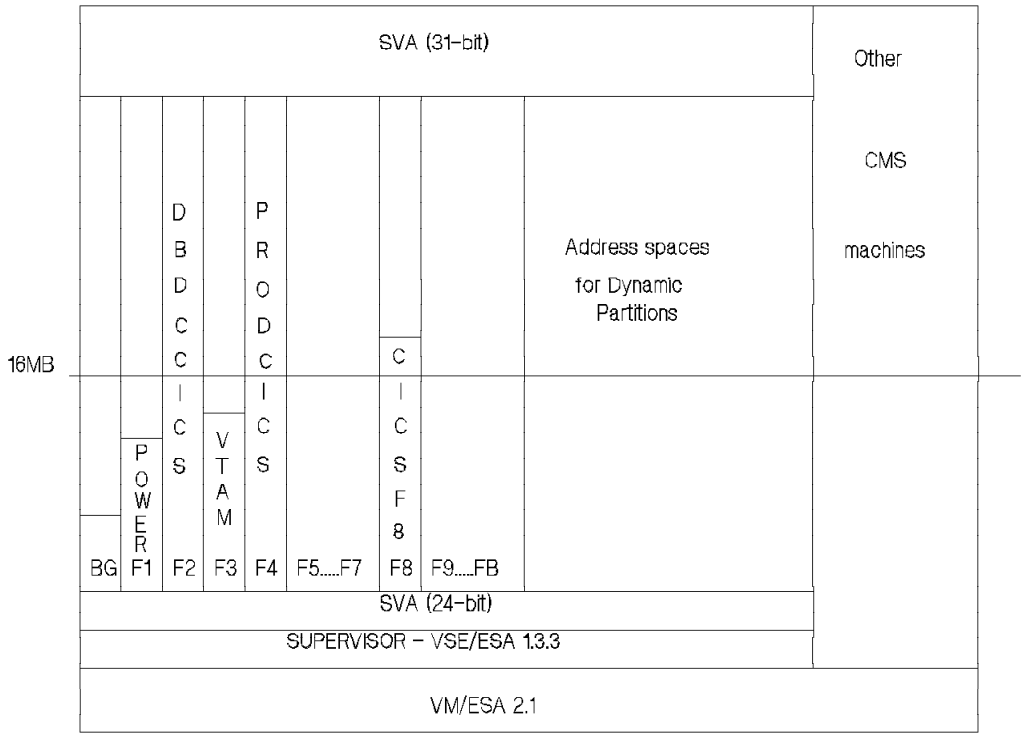


Figure 5. The ITSO Boeblingen Partition Layouts

Notes:

1. DBDCCICS runs in F2 with a partition size of 30MB
2. PRODCICS runs in F4 with a partition size of 30MB
3. CICSF8 was run in partition F8, using the default allocations

   This means that 1MB was allocated above the 16MB line, which was acceptable because of the low load on this partition. In most operational environments this would incur unnecessary overheads.

3.1.2 Application Needs

Two CICS partitions within the same VSE allow two separate ezBRIDGE Transact on VSE/ESA for MQSeries systems, while minimizing the complexity of the network definitions. The two ezBRIDGE Transact on VSE/ESA for MQSeries partitions each have their own configuration file and queue manager. Apart from the CSD, there is no sharing of data between the CICS partitions.

Each partition was designed to have two application functions, one writing to a queue for transmission to the remote system, and the other reading from a queue in the remote system. This layout is shown in Figure 6.

We used application to mean application programs. It is also possible to view an application as a collection of application programs, in which case you will define fewer applications which are likely to have access to more than one queue each.

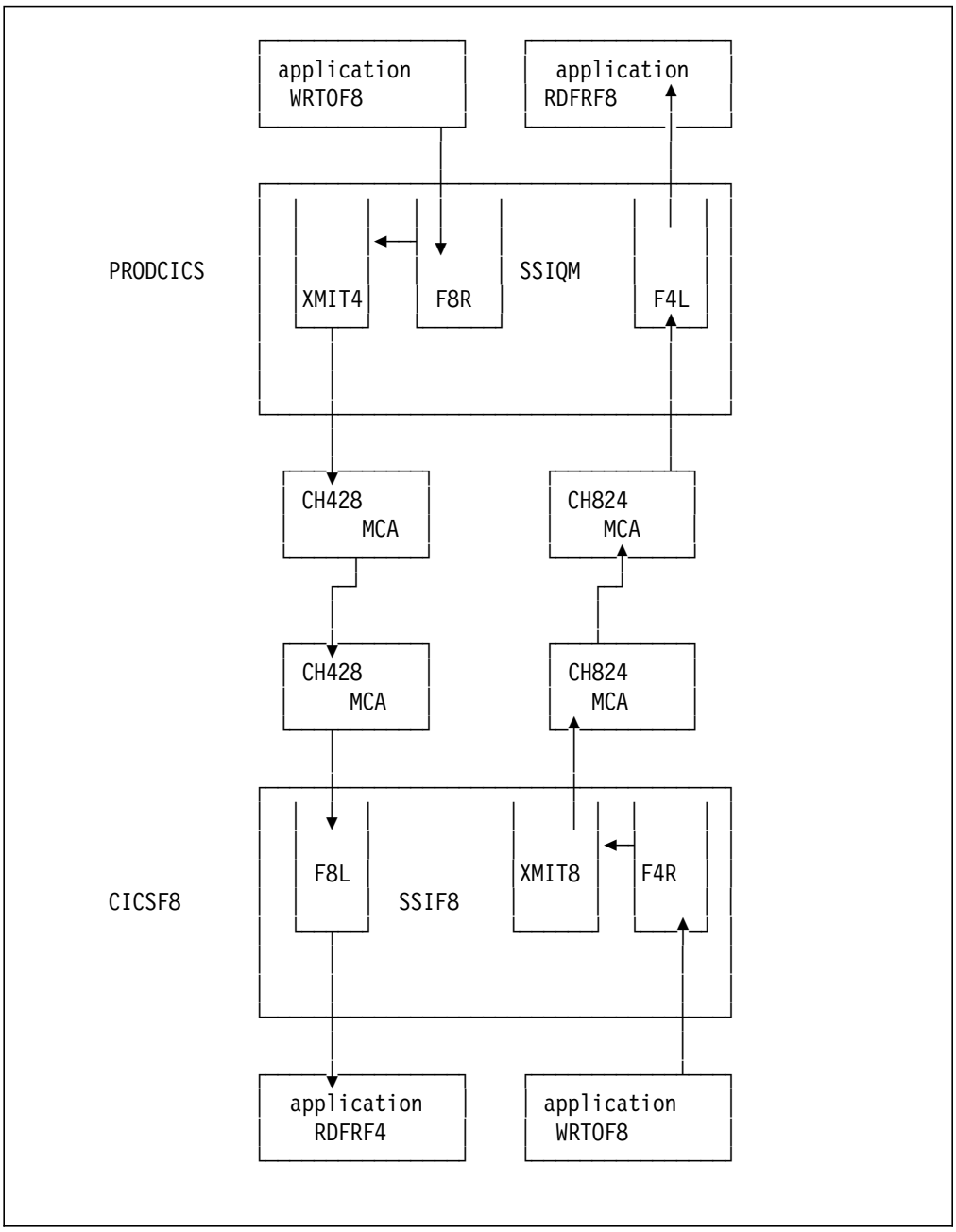


Figure 6. The Application Layout

3.1.3 Use of Planning Worksheets

Use of the worksheets in the ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide greatly assists in planning the resource definitions. Completed worksheets for this application requirement are shown in Figure 9 on page 32 to Figure 20 on page 44.

It also helps to understand the purpose of the worksheets, which is to describe the components of the system, and decide what definitions are needed to establish the desired network.

3.2 Installing the Product

Ensure that there is enough space in any existing library to which the software will be cataloged, or if it is to be put in space that is being specially allocated, that there will be enough spare space for the software and for any applications to be added to exploit ezBRIDGE Transact on VSE/ESA for MQSeries.

3.2.1 Library Definition

Little needs to be added to the instructions given in the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide, although some of the documentation is misleading. If the instructions are strictly adhered to, and the libraries for ezBRIDGE Transact on VSE/ESA for MQSeries are to be allocated in VSAM space owned by a separate user catalog, the catalog and its space must be defined before attempting to restore the libraries. The library needs to be defined, but the definitions for the sub-library are redundant, as the restore implicitly defines the libraries. The library data set definition job should be updated not only to achieve this, but also to point to the catalog by file-id and not by its DDname.

The suggested job-stream for software library restore should be amended - the first line has two POWER JECL statements in it, and the // EXEC LIBR statement should be split between LIBR and RESTORE.

Sample jobs to define the library and to restore it are provided in Figure 21 and Figure 22.

3.2.2 Data Set Definitions

The ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide advises that data sets used for queues should not be shared in any system which is not low-activity. If you follow this advice, which we recommend, then the sample definitions used in ALCQUEU.USER may be expanded. If there is to be a large number of local queues you may wish to change the naming convention to one that documents the purpose of the data sets better. Since the naming convention assumed in the installation chapter of the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide assumes a data set name structure of the form **MQSERIES.MQFI001** or **MQSERIES.MQFO001** and the second element also corresponds to the file name, you may also wish to change the filename (and therefore the FCT resource name) for consistency. However remember that if you wish to change the filename, then the data set definitions, FCT and CICS startup JCL must be kept in synchronization.

3.3 Definitions for CICS and VTAM

The ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide lists CICS resources which must be defined. The easiest method for both PCT and PPT is to assemble tables which only contain the ezBRIDGE Transact on VSE/ESA for MQSeries transactions or programs, and then use DFHCSDUP to migrate those tables to the CSD. Sample job-streams are documented in Figure 23 on page 47 through Figure 25 on page 48.

Sample definitions for LU6.2 connections and sessions are also provided in Figure 26 on page 49 through Figure 31 on page 52.

Other resources must be defined in tables, using the sample definitions provided in MQMUSR1.USER, with the member names of CICSDCT.USER and CICSFCT.USER.

The CICS startup JCL should be modified by the addition of the JCL contained in CICSJCL.USER in MQMUSR1.USER, or by a modified version of that JCL.

You are told in the ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide that the initialization program MQPINIT1 may be specified in the PLTPI, and that the shutdown program MQPSTOP can be placed in the PLTSD. We strongly advise putting both in the appropriate PLTs, MQPINIT1 for considerable operational ease, and MQPSTOP because ezBRIDGE Transact on VSE/ESA for MQSeries, when started has a long running transaction MQSM, and unless MQSM has been terminated a normal CICS shutdown is not possible.

Since the CICS definitions require parallel sessions, the VTAM application definitions were expanded to specify parallel session support. The definitions used are shown in Figure 32 on page 52. Our environment did not require external definitions, so no cross-domain resource definitions are provided. You should use the CRTE transaction or a remote transaction definition to verify that the connection and sessions can be used. We found that maximum session definitions were not adequate and therefore increased the number of sessions, and also the number of session winners on both sides. The values chosen were much larger than we anticipated we would need - we recommend deliberately setting them high, and then tuning downwards from the CICS statistics.

3.4 Definitions of ezBRIDGE Transact on VSE/ESA for MQSeries Resources

Clear guidance is given in the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide, and this is more than sufficient to define individual resources. However, there are some instances where the terminology, being specific to ezBRIDGE Transact on VSE/ESA for MQSeries, may cause some confusion. The resources used in our sample implementation are shown in Figure 33 on page 53 through Figure 44 on page 60. These definitions should be viewed in conjunction with Figure 6 on page 13. Note that in most of the definitions we used, values relating to retries, for example, have been reduced. The intention was to check the basic functional definitions.

When defining queues, you should remember that the maximum message length is the maximum record size of the file holding the queue, minus the length of the message header. Thus with a CIsize of 4096, as is used in the definitions in ALCQUEU.USER, the maximum record size is 4089, which in turn limits the length of the user part of the message to 3353 bytes, so that this is the maximum message length which may be specified.

3.5 Local Verification

The process of verification of local queues is documented in the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide, and should be followed, with at least one change. The change is that in step 5 you are told to specify that the Maximum Message Length is 4096. This exceeds the capacity of the CI, and you will be given a message. This message is an error, not a warning, and terminates the verification process. You should use a value of 3353 for this variable.

When using the TST2 transaction to test local access, you should be aware that the input is highly positional. Thus, to request writing 10 records to ANYQ, the command is as follows

```
TST2 PUT  10 ANYQ
```

and there MUST be one space between TST2 and PUT, two between PUT and 10 and one between 10 and ANYQ.

Since you will usually not have previously executed MQMT, the Global Systems Definition will need to be completed before installation verification. You would be wise to consider this an essential part of the IVP itself.

3.6 Remote Verification

Remote verification of our sample remote implementation should follow these steps. If you are verifying your own network definitions, then naturally you must change the resource names to be consistent with your own requirements.

1. Carry out local verification on both systems
2. Ensure that the connections and sessions are started on both systems
   a. On PRODCICS, this is done either by having added the group to the start list for PRODCICS before initializing PRODCICS, or by entering, from a blank screen on PRODCICS, CEDA INS GROUP(CICSF4)
   b. On CICSF8, this is done either by having added the group to the start list for CICSF8 before initializing CICSF8, or by entering, from a blank screen on CICSF8, CEDA INS GROUP(CICSF8)
   c. On PRODCICS enter CEMT I CONN
   d. Change the REL value to ACQ

      It will not change until this is also done on the other system
   e. On CICSF8 enter CEMT I CONN

      Be careful, once you have set up routing, to use a transaction which you are allowed to access (for example, avoid CEMT which has a security key of 64)
   f. Change the REL value to ACQ
   g. Verify that the status on both systems is now ACQ
3. Use CRTE from PRODCICS to verify the connection
4. On both systems, create the sample definitions from this publication
5. Ensure that all local queues are empty
6. Start ezBRIDGE Transact on VSE/ESA for MQSeries on both systems

   If the startup has been automated, you should verify at this point that both systems are active
7. Explicitly start the channels
8. On PRODCICS, enter

   TST2 PUT 05 F8R

   Remember that this is positional
9. On CICSF8, enter

   TST2 PUT 06 F4R

   Remember that this is positional
10. Use the Monitor Queues function in PRODCICS to check that 6 records are now on F4L
11. Use the Monitor Queues function in CICSF8 to check that 5 records are now on F8L
12. If records are on both local queues, then

Congratulations

You have done it

13. If they are not there you should

    a. Enter the Monitor Channels function on PRODCICS and verify that the channels are active

    b. Enter the Monitor Channels function on CICSF8 and verify that the channels are active

    c. On PRODCICS return to the Interactive Interface, enter fastpath 42 (View Message Log) and browse the message log analyzing messages

       A sample from a successful execution is provided in Figure 36 on page 55.

    d. On CICSF8 return to the Interactive Interface, and enter fastpath 42 (View Message Log) and browse the message log analyzing messages

       A sample from a successful execution is provided in Figure 39 on page 57.

    e. Correct and retry

3.7 Messages

You may receive some messages which are not among those listed in the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide. Since there is no detailed explanation of the messages, the message text is intended to be self-explanatory.

We recommend that you read any messages carefully, to be sure that you understand their intention.

3.8 Application Programming

The API is best learned by studying the sample applications delivered with the product, in conjunction with the ezBRIDGE Transact on VSE/ESA for MQSeries User′s Guide chapter on the Application Programming Interface.

3.8.1 Programming Languages

The ezBRIDGE Transact on VSE/ESA for MQSeries subroutines are written using VS COBOL II and were compiled using VS COBOL II 1.3.2. This means that they are portable to other systems (including VSE/ESA with VS COBOL II 1.4). As access to the MQI is through static calls, the calling program must be written in VS COBOL II, so you must have at least the VS COBOL II run-time libraries on the ezBRIDGE system. You will also need the compiler itself either on that system, if it is used for development, or on a separate development system.

3.8.1.1 Applications not written in COBOL II

The ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide shows two methods of implementing MQSeries applications written in programming languages other than COBOL II. Since both techniques depend on calls, either to the MQI or to programs which themselves access the MQI, these techniques will not work.

ezBRIDGE Transact on VSE/ESA for MQSeries can only run in a CICS/VSE environment, and in the implementation current at the time of writing this manual, that is, CICS/VSE 2.2, under CICS/VSE a COBOL II program may not be called by a program written in any other language. Thus to use another language, the program must issue an EXEC CICS LINK or EXEC CICS XCTL command to link, or transfer control, to a program written in VS COBOL II which will in turn issue the static calls to the ezBRIDGE Transact on VSE/ESA for MQSeries interface programs, which are themselves written in VS COBOL II. This would mean that the application would need to be a mixture of COBOL II and the other language, with the actual calls to the MQI issued from COBOL II. The technique of using EXEC CICS LINK from an assembler program to a COBOL II program is the method we used in our sample batch implementation.

3.8.2 The Sample Programs

The sample programs carry copyright notices, and are intended for installation verification and for providing examples of coding sequences and techniques. To generate consecutive code some copy and include statements have been resolved by directly inserting the code concerned. In at least one case a statement has been omitted which does not affect the execution of the sample, but which, if omitted in another context, would cause difficulties.

3.9 31-bit Exploitation

Most of the subroutines supplied with ezBRIDGE Transact on VSE/ESA for MQSeries have residency and addressing mode attributes which will allow 31-bit exploitation. One however does not, and this is MQPEIB1, which is always included. Since ezBRIDGE calls are static calls, this means that 31-bit exploitation is not possible, unless the application is split and uses either EXEC CICS LINK or EXEC CICS XCTL commands from a 31-bit enabled program to a general interface which is link-edited as 24-bit.

3.10 ASCII-EBCDIC Conversion

CICS/VSE users with applications communicating with other CICS family platforms which use ASCII will be familiar with the fact that conversion tables can be defined which will cause automatic conversion between ASCII and EBCDIC and between EBCDIC and ASCII. For CICS/VSE the rules and implementation are documented in *Communicating from CICS/ESA and CICS/VSE*, SC33-0825. The MQSeries however, has no such capability, and currently ASCII-EBCDIC and EBCDIC-ASCII conversion is seen as an application responsibility. This is no problem for our sample remote implementation, where no conversion is required, but in a mixed network is of great importance.

Chapter 4. A Sample Batch Interface for ezBRIDGE Transact for VSE/ESA

The interface is designed to extend the power and flexibility of the Messaging and Queuing Interface to batch applications as well as on-line applications in the VSE/ESA environment. This was necessary because VSE/ESA is the only platform for the MQI which does not have a batch interface provided as part of the product.

4.1 Design Considerations

The design criteria for the interface were carefully considered and the following conclusions were reached.

- The length of each record written to a queue is limited by the CI-size of the data set on which the queue is defined. This can easily be changed by deleting and redefining the VSAM data set with a larger CI-size. However, to increase the maximum size of the records to be written would also require changes to the interface programs to ensure that the buffers and data structures in the programs were large enough to allow the desired record sizes.
- Typically you will want to be able to use a CICS/VSE to host a mixture of work, including the server task for the batch interface, CICS applications using the MQI, and also traditional CICS transactions which have no access to the MQI. This means that we had to design and implement a method in which no CICS task issues a wait on a non-CICS event. This clearly means either that no wait on a non-CICS event can be allowed at all, or if one is considered unavoidable, it must be issued from a VSE sub-task to avoid any impact on CICS.
- We carefully considered the relative merits of using XPCC or LU6.2 data streams, and decided to proceed with the XPCC design. An LU6.2 implementation in batch would give much better performance, as it would not limit throughput in the same way as our main-task to sub-task implementation. However since VSE does not have a native APPC, this would have been a much more complex design and implementation.

Therefore the following decisions were made and implemented.

- We designed the CICS task as a long-running transaction. That is, it must be started before the first batch transfer and will run until explicitly shut down. Its initialization could be started from the PLTPI, but this would also require that ezBRIDGE Transact on VSE/ESA for MQSeries initialization and batch interface initialization are serialized and coordinated.
- The batch subroutine copies the MQ data out of the calling program′s working storage into the XPCC buffer before it sends the data to the CICS main task program. Conversely, the batch interface has to copy the MQ data out of the XPCC buffer into the batch calling program′s working storage before passing control back to the calling program.
- Data sizes have to be static and known in advance to prevent problems with misalignment of data as it is passed from one program to another.

- The only way to prevent a WAIT on an XPCC ECB from causing the whole CICS partition to WAIT, is to ATTACH a sub-task to the CICS task and allow the sub-task to do all the XPCC work necessary on the CICS side, including WAIT processing.
- Because the sub-task runs under its own TCB, it does not have access to CICS services, we therefore had to cause the CICS main task to wait via the EXEC CICS DELAY service. Since the shortest CICS delay is one second, this limits the throughput of the interface to one record per second.
- To synchronize activity between the main task and the sub-task requires the use of a semaphore or “pseudo ECB” as it is called in the program listings.
- Synchronization between the sub-task and the batch subroutine is achieved by conventional use of XPCC ECBs.

The objective of the project was not to write a general purpose interface between a batch application and an ezBRIDGE Transact on VSE/ESA for MQSeries Queue Manager, but rather to validate the design of a mechanism for the interchange of data between a batch job and a CICS partition, and to confirm its validity for passing some MQI requests. Depending on the application requirements, this may be sufficient. Some cases will certainly need modifications to one or more components of the sample interface, but there are enough comments provided to make this relatively simple.

4.2 Components of the Interface

The interface consists of the following six programs, one of which is a sub-program:

EZBMBTCH Assembly language subroutine of a batch program that does the XPCC transfer of data to and from the CICS task. Written in Assembler because the use of XPCC requires Assembler.

EZBMTASK Assembly language CICS program that ATTACHes a sub-task to do the XPCC data transfer to and from the batch task, and also LINKs to MQICICS, which does the actual ezBRIDGE Transact calls. Written in Assembler because the use of XPCC and the ATTACH macro requires Assembler.

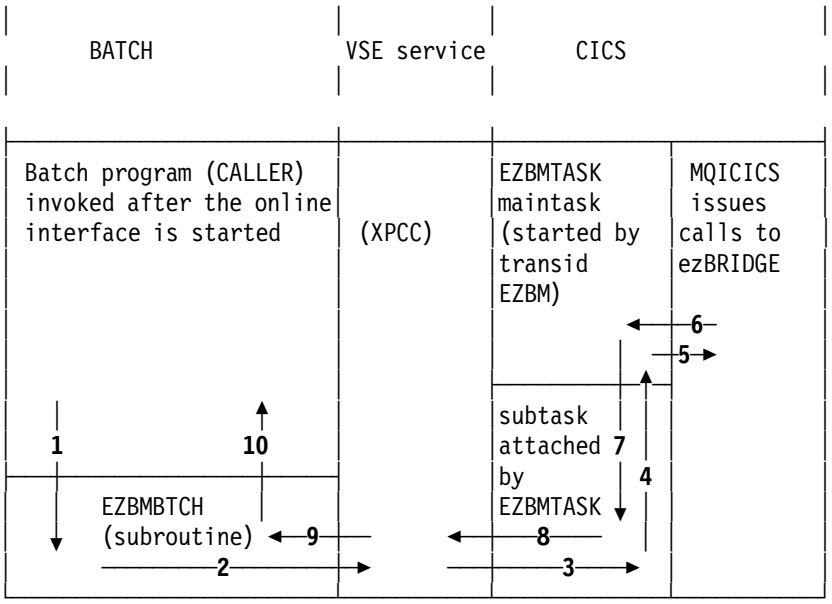
MQICICS COBOL II program that receives the data from EZBMTASK and does the CALLs to the specific MQ program (for example MQOPEN). This program then passes the received data back to EZBMTASK. This program was written in COBOL II because the MQ programs are written in COBOL II. COBOL II programs cannot be called from other languages than COBOL II under CICS.

CALLER COBOL II program that submits send and receive requests to EZBMBTCH. This program was modelled on the test program that is shipped with ezBRIDGE Transact for VSE/ESA called TTPTST2.

TASKCAN Assembly language program that from batch issues an XPCC request to terminate XPCC under CICS and end both the CICS main task and sub-task.

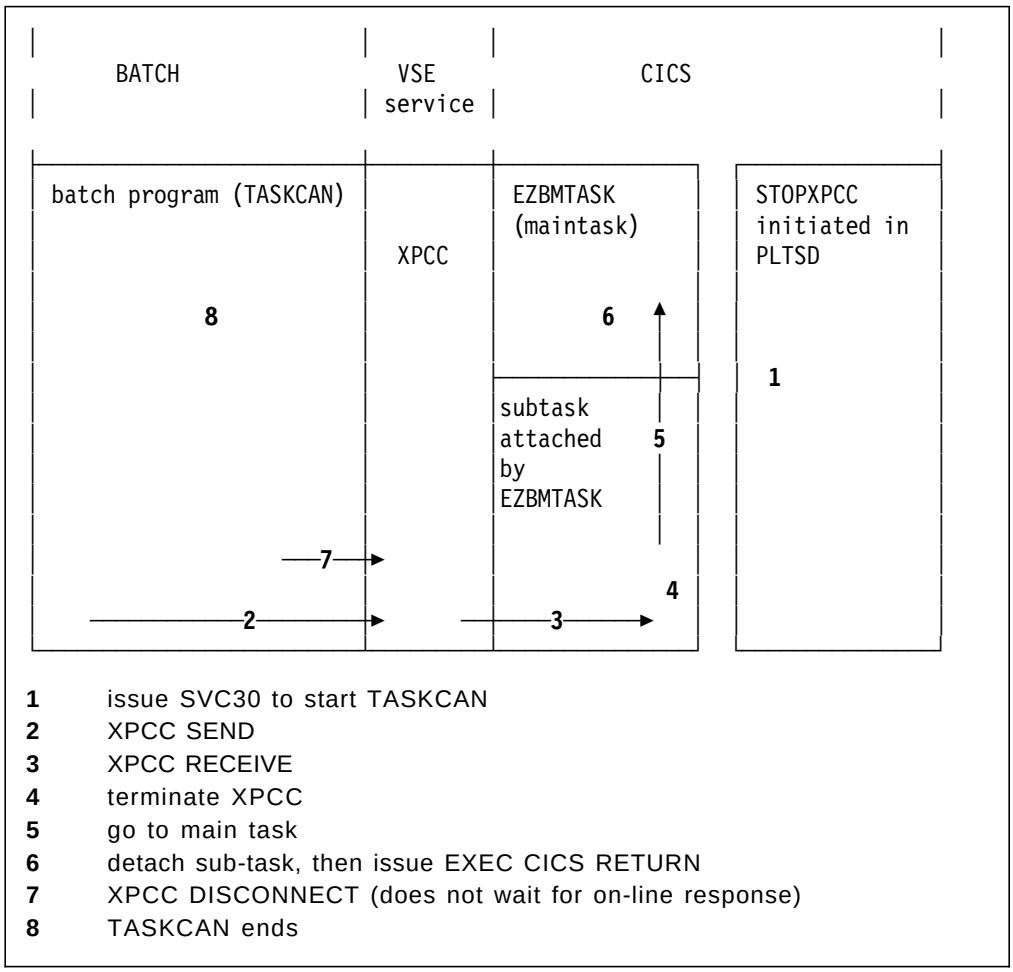
STOPXPCC Assembly language program that releases the job that runs TASKCAN. This program should be specified to run in the CICS PLTSD.

A schematic of the components of the interface used for normal running is shown in Figure 7 while the components used to shutdown the system are shown in Figure 8.



1 Call EZBMBTCH
2 XPCC SEND
3 XPCC RECEIVE
4 post received for main task
5 LINK to MQICICS
6 RETURN to main task
7 post ready to send for sub-task
8 XPCC SEND
9 XPCC RECEIVE
10 return to CALLER

Figure 7. Components of the Batch Interface - Normal Running. The arrows indicate major flows of requests, and the key below shows what function is being requested.



1 issue SVC30 to start TASKCAN
2 XPCC SEND
3 XPCC RECEIVE
4 terminate XPCC
5 go to main task
6 detach sub-task, then issue EXEC CICS RETURN
7 XPCC DISCONNECT (does not wait for on-line response)
8 TASKCAN ends

Figure 8. Components of the Batch Interface - Automatic Shutdown. The arrows indicate major flows of requests, and the key below shows what function is being requested.

4.3 Assumptions and Restrictions of the Interface

We made a number of assumptions about the probable use that would be made of the interface. Our implementation dictated a small compromise in the design. This is documented in the list below.

- Maximum XPCC record size was restricted to 4KB. This can be increased. However, if this is done, the sizes of the data structures in all four of the interface programs used in normal running must be changed to allow the use of bigger buffers. The VSAM data sets containing the queues must also be recreated with a big enough CI-size to accommodate the larger record size. A queue data set with a CI-size of 4096 bytes allows the use of queue records of up to 3353 bytes. We also decided to limit the size of the message buffer to 516 bytes. This can be overcome by increasing the size of the data structures in the calling program and the MQICICS program, as well as changing the maximum buffer length in the EZBMBTCH program. Our implementation used a maximum message size of 516 bytes because that is the maximum size used by TTPTST2.
- Only the most basic Message Queuing calls were catered for. They are:

  MQCONN

MQOPEN

MQPUT

MQGET

MQCLOSE

MQDISC

There is no provision in our interface for MQPUT1 or MQINQ, but support for these two calls could be added quite easily.

- There are two new calls added, one to allow the batch subroutine and the CICS routine to terminate their XPCC session without causing any abnormal error conditions, and the other to cause termination of the long-running CICS task. They are explained in the next section.
- Data conversion is not catered for in the interface, because we were not sending data to or receiving data from ASCII platforms.
- There is an inherent limitation in the design that causes the message throughput to be no more than one message record per second. This is fine for small amounts of data, but may mean that the application and message design need to be changed when large amounts of data are to be read or written.
- Our solution was not conceived as a complete solution to all business problems, but should be seen as a sample which can be used as it is documented for those tasks whose needs it meets exactly, but which can also be extended by minor changes to meet individual needs.

  An example is that although the implementation limits throughput and therefore is suitable for relatively small volumes in terms of number of messages, the MQI is directed at asynchronous transfers, so that transfer rate may not be critical. Where it is, then packaging several messages together will allow more than one logical message per second to be transferred.

4.4 Running the Sample Batch Interface

We tried to make the calling conventions used between the batch calling program and the batch subroutine as similar as possible to those required in an on-line program. The only difference is the MQ service to be called is passed as the first parameter with a length of eight bytes, and the program called is always EZBMBTCH.

Anybody reading the source code of EZBMTASK and EZBMBTCH will notice the large number of WTO (write to operator) macros contained in them. These were there for debugging purposes, but have been left in so that a user of the sample interface can easily follow the logic flow while the system is running. On an operational system these should, of course, be removed to avoid cluttering up the console with messages which are only needed during development.

The CICS transaction must be started before any jobs that access the interface are run. When you start the transaction and see the following messages on the console, the interface is initialized and waiting for input:

IN SUBTASK

IDENTIFIED

The two new calls mentioned in the previous section to disconnect XPCC and terminate the transaction are of the following form (where EZDISC and EZQUIT are eight-byte fields containing a constant of “EZDISC” and “EZQUIT” respectively):

- CALL ′EZBMBTCH′ USING EZDISC. This causes the XPCC connection to terminate and the CICS task to wait for a reconnection.
- CALL ′EZBMBTCH′ USING EZQUIT. This causes the XPCC connection to be terminated, and terminates the CICS transaction. An example of a program using EZQUIT, is the sample program TASKCAN.

  The program STOPXPCC should be included in the PLTSD to release the TASKCAN job, which will run the TASKCAN program and close the XPCC interface gracefully, as well as end the CICS transaction.

4.5 Did We Meet the Objective?

Our objective was to build an extension to ezBRIDGE Transaction that allowed us to transfer data between a batch job and a Message Queuing queue. Under VSE, these queues are only accessible from CICS.

Using the programs written, we succeeded in moving data to an MQ queue and retrieving it from the queue by means of our batch interface. However, because of design constraints, the throughput is limited.

4.6 How to Access the Source

The source code for the batch interface is documented in Appendix 5 of this manual, where instructions are provided for capturing the code from softcopy accessed by BookManager Read/2.

Chapter 5. Differences Between ezBRIDGE Transact on VSE/ESA for MQSeries and Other MQSeries Implementations

The following is a list of differences which may exist between ezBRIDGE Transact on VSE/ESA for MQSeries and other MQSeries platforms. The differences will not be constant between different platforms, and the list we provide here is of **potential** differences derived from the documentation for a number of platforms. A short section expanding on each listed difference is included. The list is not intended to be a complete list of differences, but does list everything we could identify.

- Application differences
- Languages
- Different channel types
- MCA activation
- Persistent messages
- Name lengths may differ
- Different conditions for the dead letter queue
- VSE has no documented channel trace

5.1 Application Differences

The first and most fundamental difference is that the ezBRIDGE Transact on VSE/ESA for MQSeries implementation is restricted to CICS/VSE only, with no batch interface being made available. This is, of course, the reason for this book. The User′s Guide also warns that MQI features have been implemented in a slightly different manner for different operating system environments. It urges that Chapter 7, *Application Programming Interface*, should be reviewed closely to ensure that any features of particular interest are available on VSE/ESA.

5.2 Languages

Only COBOL II is supported in a VSE/ESA environment, but this is largely because of restrictions imposed by the CICS/VSE application environment. Other platforms, where the same restriction is not imposed, allow more choice of programming languages.

5.3 Different Channel Types

Channel types supported by the MQSeries are:

1. SENDER channels

   Originates data and initiates remote RECEIVER
2. RECEIVER channels

   Receives data, initiated by SENDER
3. REQUESTOR channels

   Receives data and initiates remote SERVER
4. SERVER channels.

   Originates data, initiated by REQUESTOR

Of these only sender and receiver channels are supported in VSE/ESA and consequently, when planning a network where one of the participants is VSE/ESA it is important that you ensure that the channels at either end of a connection are not only compatible with each other, but also are compatible with the requirements of the application.

As another channel consideration, each communication channel has a transport protocol associated with it, and although a number of protocols are supported by MQSeries, the only one supported by ezBRIDGE Transact on VSE/ESA for MQSeries is LU6.2. Other protocols cannot be used by VSE, and consequently a VSE/ESA ezBRIDGE node can only be attached to another MQM node which both supports and has implemented LU6.2.

5.4 MCA Activation

MCA is the communications engine for ezBRIDGE, running the Message Channel Protocol program. The various User's Guides for ezBRIDGE on VSE/ESA, OS/400*, OS/2* and AIX* do not make clear distinctions in every case between the starting of an MCA and the starting of an MCP.

Under VSE we found it necessary, even when the ezBRIDGE system had been started (automatically or manually), to start the individual channels. The MCP programs get started as a result of trigger specifications on a transmission queue.

We recommend, if the documentation states that an MCA must be started manually, you examine the rest of the manual carefully to ensure how the MCP is started. They may be apparent differences in the documentation as well as real differences in the actual implementation.

5.5 Persistent Messages

A message is termed **persistent** if it will survive a queue manager restart. Currently ezBRIDGE Transact on VSE/ESA for MQSeries only supports persistent messages. This characteristic should be considered when planning connections to other MQSeries systems which allow this attribute selectively.

5.6 Name Lengths May Differ

Name lengths and other permitted values may differ. Specifically, the remote task ID in a sender channel definition is limited under ezBRIDGE Transact on VSE/ESA for MQSeries to four characters. When communicating with another ezBRIDGE on VSE this is no problem, as this value then should be MQ01 which is the CICS/VSE transaction ID associated with the channel receiver program. However, if the receiver is on an OS/400 ezBRIDGE system, the process name is EZRCVCL, which is the receiver command list. The question then is how you can overcome the problem of specifying a seven-character value in a four-byte field.

Fortunately this is documented by ezBRIDGE on OS/400, which tells you how to create a routing entry for a process.

1. Use the command ADDRTGE to add a routing entry for EZRCVCL
2. In this define the program as EZRCVCL

3. Define the Library in which it will be found
4. Define a compare value of 'EZRC'

   The compare value must be in apostrophes, and the value between the apostrophes is a four-byte value which can be use for the Remote Task ID in the ezBRIDGE Transact on VSE/ESA for MQSeries Channel Record definition.

5. The resulting entry is as follows:

```
Seq Nbr      Program      Library      Compare Value      Start Pos
             EZRCVCL      SSIMQILIB    'EZRC'             37
```

5.7 Dead Letter Queue

The ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide only documents the use of the Dead Letter Queue in the following cases:

- Wrong Queue Manager Name
- Wrong Queue Name

ezBRIDGE for OS/400, AIX/6000* and OS/2 use the Dead Letter Queue under the following circumstances:

Requestor/Receiver Channels

- The remote queue is full
- The remote queue is PUT inhibited
- The message sent to the remote queue is too large
- The remote queue does not exist
- The message contains a duplicate Message Sequence Number

Sender/Server Channels

- The remote queue manager uses the Dead Letter Queue
- The message on the transmission queue is greater in size than the Maximum Message Size

We were not able to try and force some of the other circumstances for use of the Dead Letter Queue under VSE/ESA, however, and must point out that the documentation of the other platforms does include separate sections on the Dead Letter Queue. It is possible that the DLQ is, in fact, used in more cases by ezBRIDGE Transact on VSE/ESA for MQSeries, but that these are not documented.

5.8 Trace at Channel Level

VSE/ESA MQSeries does not have trace at Channel Level.

AIX/6000 and OS/2 MQSeries allow an operator to start or stop the Channel Trace. The trace entries are written in the error log. The error log should be examined to assist in trouble shooting communication problems. The operator has an option to select the channel to be traced.

Appendix A. Planning Worksheets Used in the Sample

The worksheets completed for the sample remote definitions are given below. Our experience was that it was significantly easier to fill in the worksheets properly after using them for the first time. For this reason further guidance on completing them is given here.

Note that some worksheets are used to document the characteristics of the system or network, but usually result in naming an MQSeries resource. As you progress through the worksheets you will finally arrive at the names of the resources to be defined to ezBRIDGE, with an indication of their relationship to each other. This is contained in the channel list worksheet and the routing table.

A.1 System List - Worksheet

One list should be provided for the entire network, identifying every system which will use messaging and queuing. This worksheet is used to allocate the message queue manager name, which will be needed not only in that system, but also in any other system which defines a remote queue which is local to that queue manager.

System	Location	Hardware	Msg_Q_Mgr	Comments
PRODCICS	Boeblingen	9121/VSE	SSIQM	F4 on V131A90
CICSF8	Boeblingen	9121/VSE	SSIF8	F8 on V131A90

Column 1 System name or identification (user specified - note that we used the applid of the CICS/VSE system). Every MQSeries system in the network installed should be listed. Where, as in our case, more than one instance of an MQSeries product is installed, there must be an entry for each one. Thus we have ezBRIDGE installed in each of two CICS/VSE partitions, so each one is shown separately.

Column 2 Location of system. This helps in the identification of the system.

Column 3 Type of hardware. If the hardware does not imply a particular software platform, you would be well advised to indicate it here. This is important as it will be needed to identify characteristics used for some object definitions. For example, in a Channel Record, you need to identify the channel receiver at the other end. This will depend on the platform used.

Column 4 Assigned message_queue_manager name. This is the primary output from this worksheet, and each line should be assigned a value which is unique in the system. This is the name used for the system when creating the Global System Definitions for that system.

Column 5 Any user comments

Figure 9. System List Worksheet Completed for the Sample

A.2 Application List - Worksheet

One list is completed for the entire network, identifying all applications which will use messaging and queuing. Each is assigned at least one local queue name through which it will receive messages. The local queues used by the applications are mapped to the host systems from Figure 9 on page 32 by means of the *Msg_Q_Mgr* field. The new element in this worksheet is the application name, so that is the output from this worksheet. There must be at least one application for each queue manager in the network, but there will usually be more.

Which applications?

The ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide suggests using this only for applications which receive data. We found it more useful to list all applications.

Each line in this worksheet should be used to build a separate Application Look at Queues Worksheet.

Application	Queue_Name	Hardware	Msg_Q_Mgr	Comments
RDFRF4	F8L	9121/VSE	SSIF8	
RDFRF8	F4L	9121/VSE	SSIQM	
WRTOF4	F4R	9121/VSE	SSIF8	
WRTOF8	F8R	9121/VSE	SSIQM	

Column 1 Application name. Here we used the name of the application program concerned.

Column 2 Queue name. Only identify queues defined locally.

Column 3 The hardware (and SCP) used should be identified.

Column 4 The queue manager of this system is identified.

Column 5 Put any user comments here.

Figure 10. Application List Worksheet for the Complete System

A.3 Application Look at Queues - Worksheet

One list is to be completed for each application. The new element here is that the locally defined queues are related to the applications. Since the application is named in the uppermost box, this shows which application is being described. For clarity the system or queue manager should also be shown.

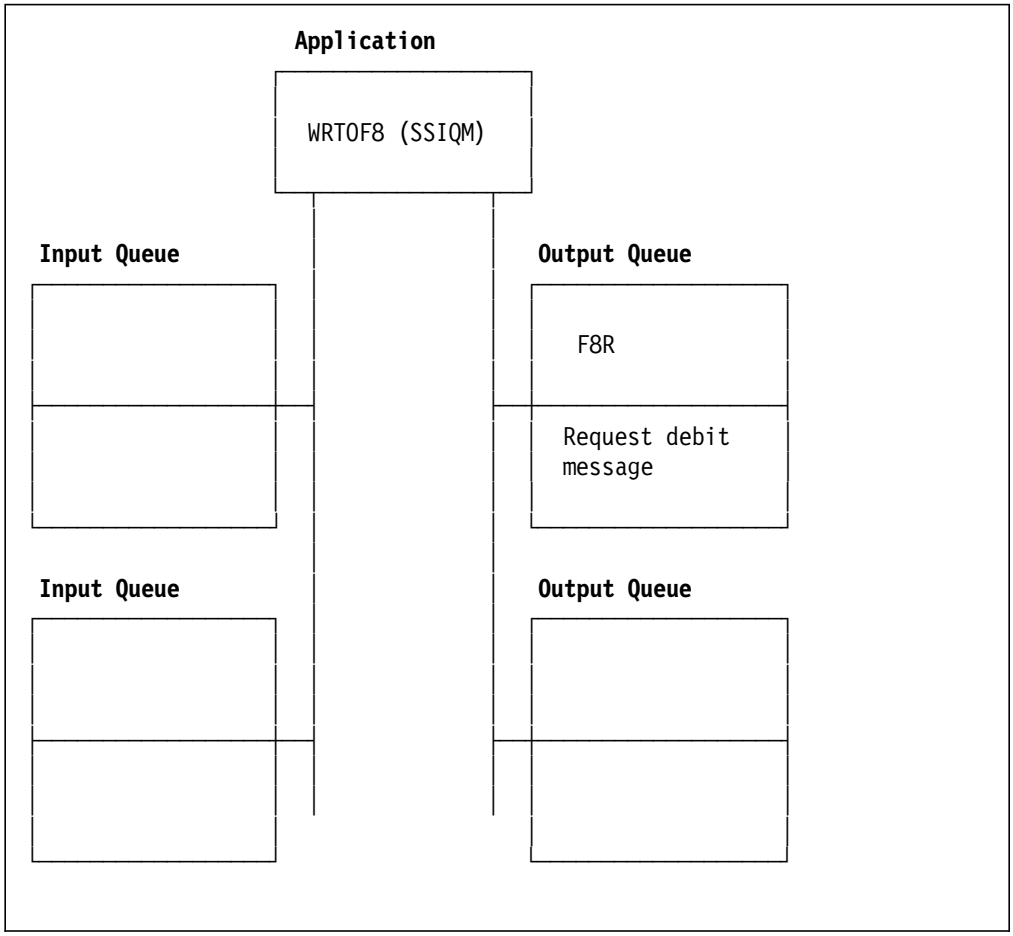


Figure 11. Queue Access for the WRTOF8 Application. This application writes to F8R.

Each queue box contains the queue name, and a short description of the message format. This can be either a brief text description, or it could be the name of the COBOL structure used.

Each application must access at least one queue. Some applications may interact with more than one queue, and if so all queue interactions must be documented.

One list is to be completed for each application. The new element here is that the locally defined queues are related to the applications. Since the application is named in the uppermost box, this shows which application is being described. For clarity the system or queue manager should also be shown.

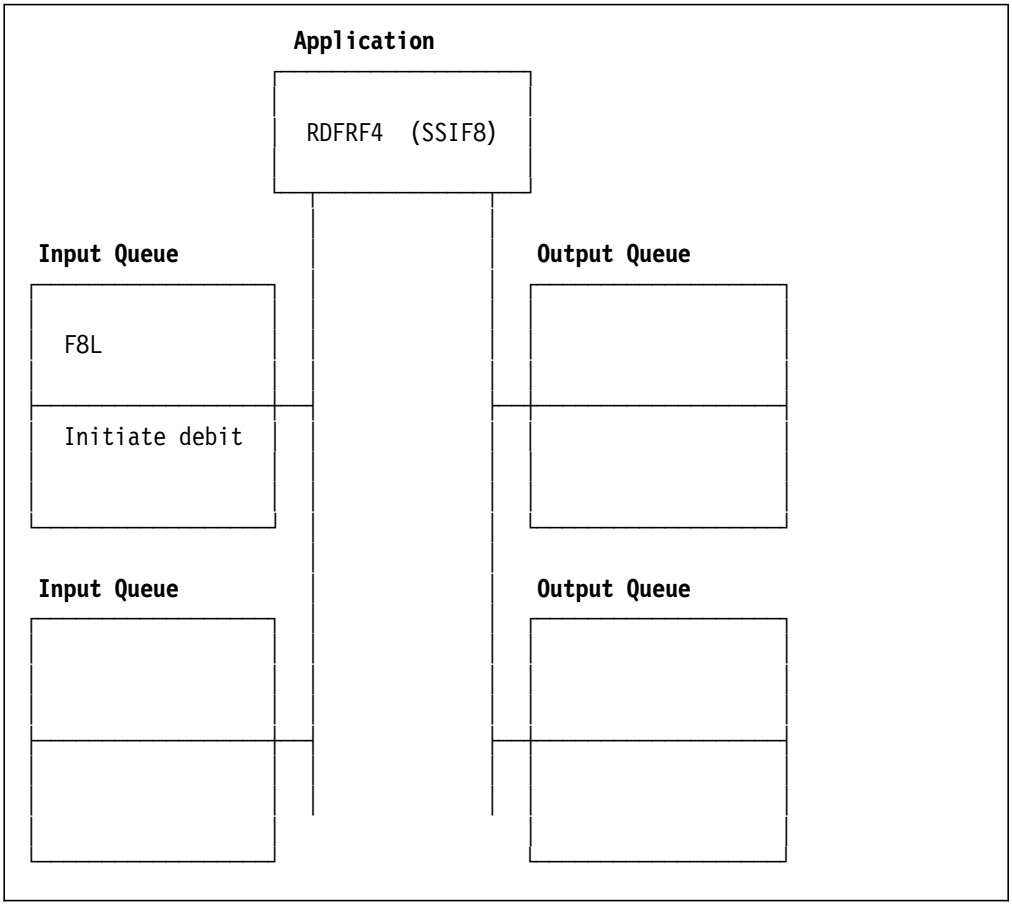


Figure 12. Queue Access for the RDFRF4 Application. This application reads from F8L.

Each queue box contains the queue name, and a short description of the message format. This can be either a brief text description, or it could be the name of the COBOL structure used.

Each application must access at least one queue. Some applications may interact with more than one queue, and if so all queue interactions must be documented.

One list is to be completed for each application. The new element here is that the locally defined queues are related to the applications. Since the application is named in the uppermost box, this shows which application is being described. For clarity the system or queue manager should also be shown.

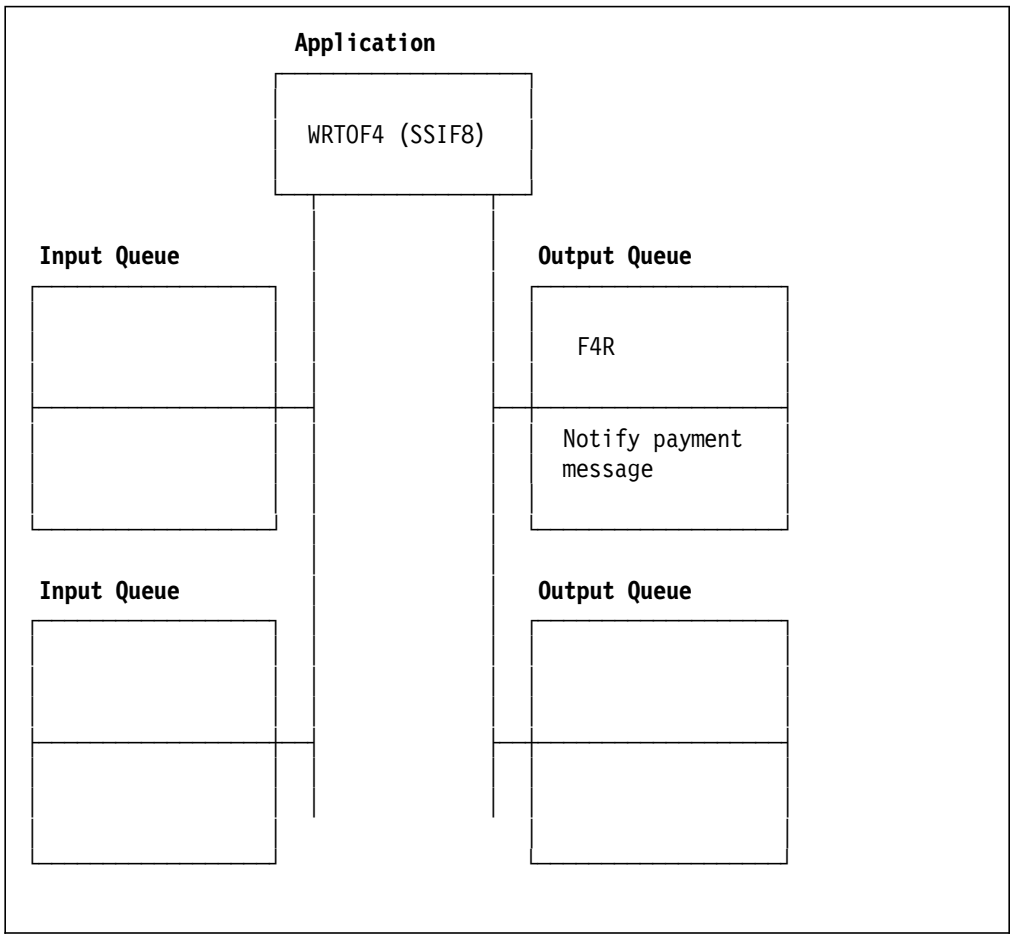


Figure 13. Queue Access for the WRTOF4 Application. This application writes to the remote queue F4R.

Each queue box contains the queue name, and a short description of the message format. This can be either a brief text description, or it could be the name of the COBOL structure used.

Each application must access at least one queue. Some applications may interact with more than one queue, and if so all queue interactions must be documented.

One list is to be completed for each application. The new element here is that the locally defined queues are related to the applications. Since the application is named in the uppermost box, this shows which application is being described. For clarity the system or queue manager should also be shown.

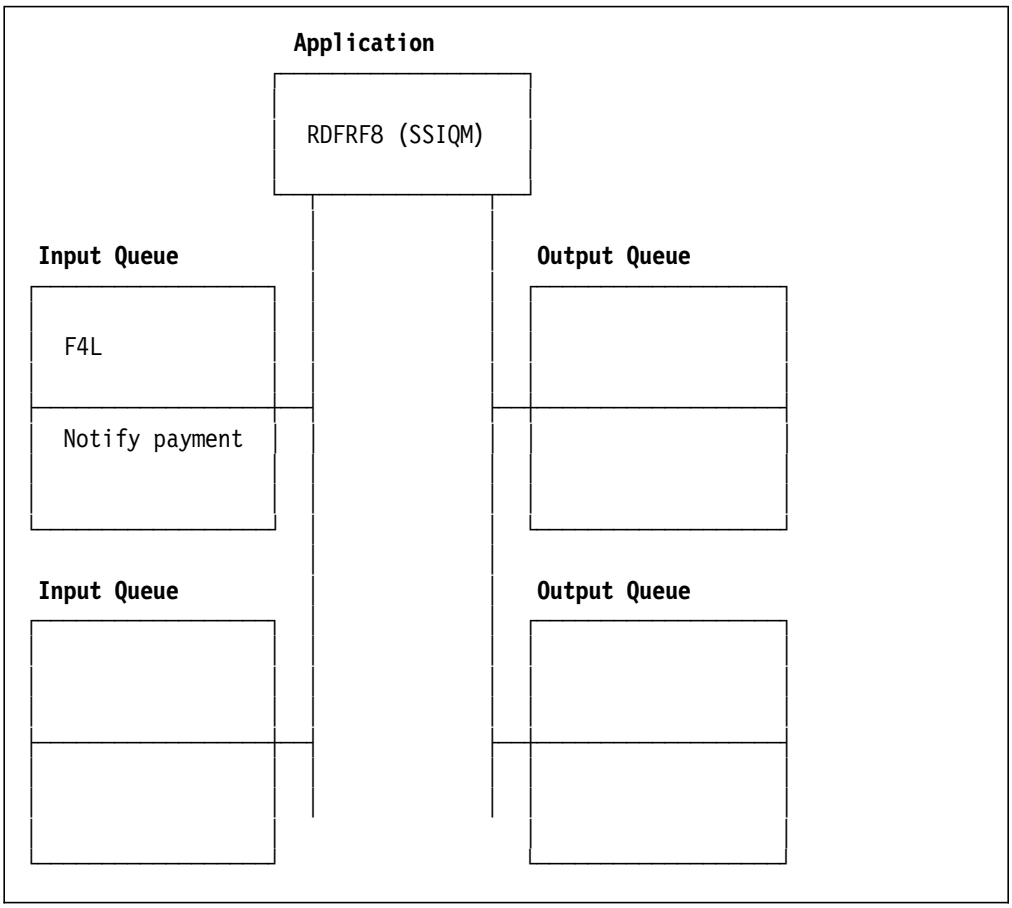


Figure 14. Queue Access for the RDFRF8 Application. This application reads from F4L.

Each queue box contains the queue name, and a short description of the message format. This can be either a brief text description, or it could be the name of the COBOL structure used.

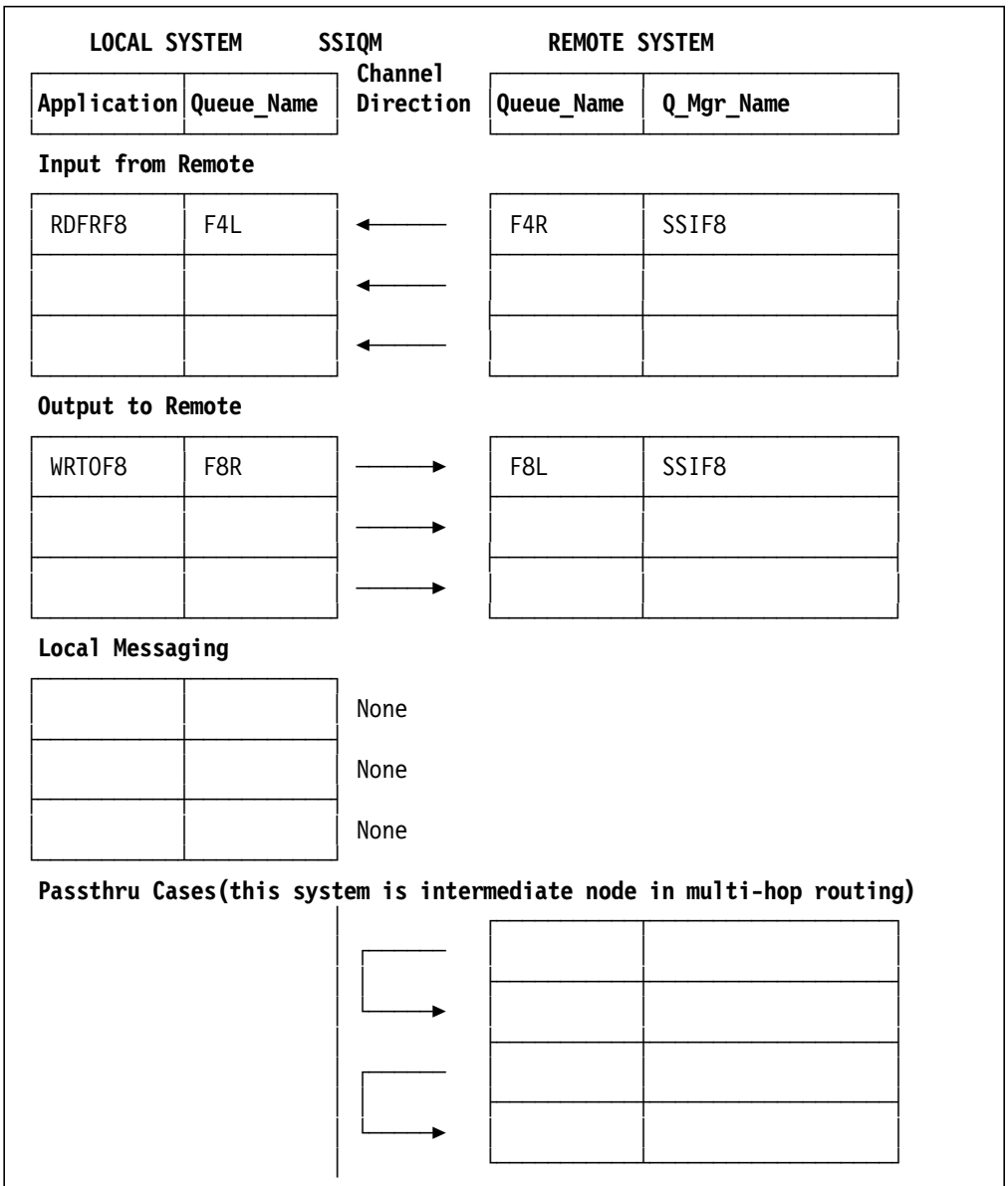
Each application must access at least one queue. Some applications may interact with more than one queue, and if so all queue interactions must be documented.

A.4 System Look at Queues - Worksheet

One list is provided for each system, and in these the relationship between applications and queues is established. Because of the way we have set up our systems, these lists are symmetrical, although this will often not be the case.

The system **must** be clearly indicated, and this is done at the head of the worksheet.

This worksheet combines the previous ones. The new element is that we are now identifying whether access is local or remote, and for remote queue access we are indicating the direction of flow, and also which applications are partners in a transmission.



Column 1 Local system application.
An entry is needed under Input from Remote, Output to Remote or Local Messaging for every line in the Application List Worksheet which is associated with this queue manager.

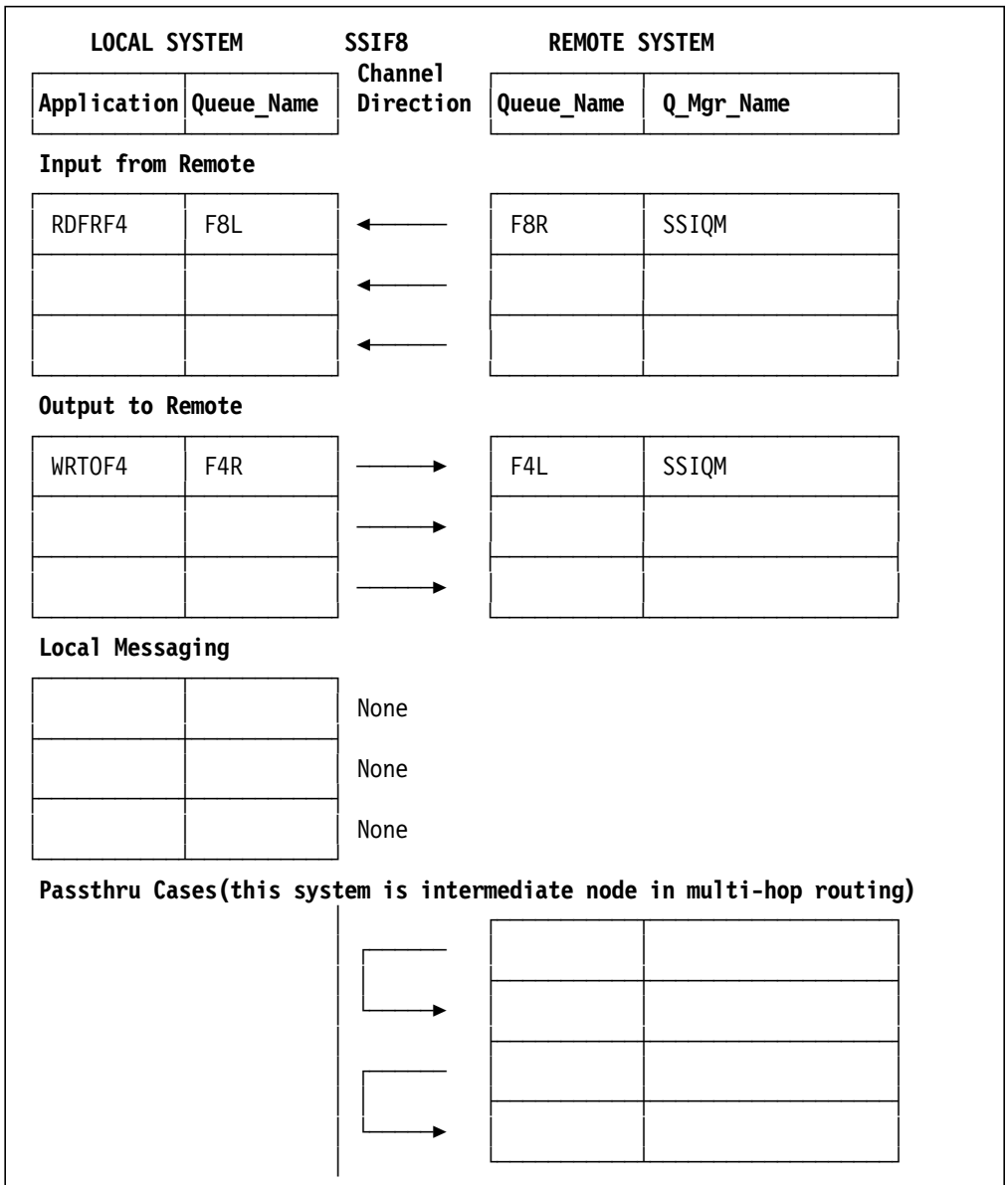
Column 2 Local system queue name

Column 3 Channel (shows the direction of message flow)

Column 4 Remote system queue name

Column 5 Remote system queue manager name

Figure 15. Definition of PRODCICS. This shows the queues on this system, with their relationship to other queues. View this together with Figure 16 on page 40.



Column 1 Local system application
An entry is needed under Input from Remote, Output to Remote or Local Messaging for every line in the Application List Worksheet which is associated with this queue manager.

Column 2 Local system queue name

Column 3 Channel (shows the direction of message flow)

Column 4 Remote system queue name

Column 5 Remote system queue manager name

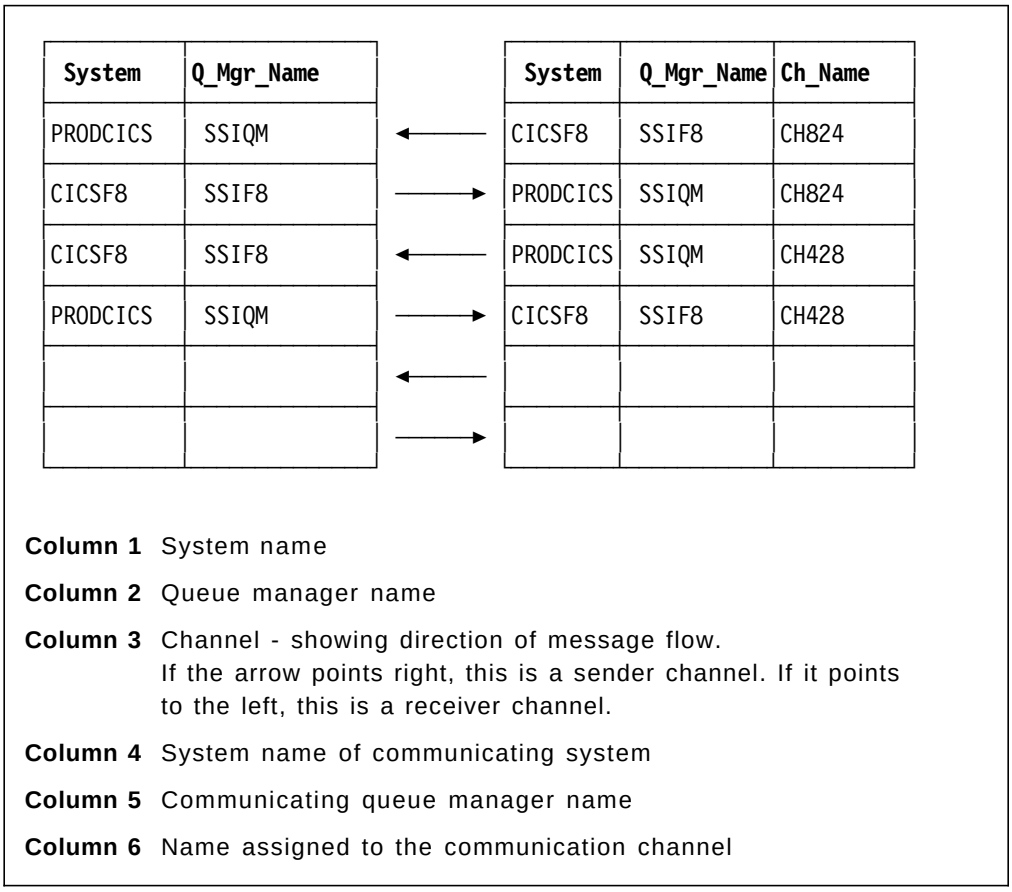
Figure 16. Definition of CICSF8. This shows the queues on this system, with their relationship to other queues. View this together with Figure 15.

A.5 Channel List Worksheet

This worksheet is used to assign names to the channels, by which they will be defined in both systems. Note that the channels are defined in pairs.

This worksheet represents the entire system.

The channel name is the name of the MCA. Although there appears to be no such restriction documented in the ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide, the corresponding manual for ezBRIDGE on OS/400 clearly states that *The channel name must be identical on both systems, including case*. To design for greatest flexibility, if a restriction is stated for one platform, it should be observed also on the others. Each channel **must** have two lines in this worksheet, since each line will result in a channel definition. The final column shows the channels which must be defined, and the previous column shows which queue manager must be used when defining the channel.



System	Q_Mgr_Name		System	Q_Mgr_Name	Ch_Name
PRODCICS	SSIQM	←	CICSF8	SSIF8	CH824
CICSF8	SSIF8	→	PRODCICS	SSIQM	CH824
CICSF8	SSIF8	←	PRODCICS	SSIQM	CH428
PRODCICS	SSIQM	→	CICSF8	SSIF8	CH428
		←			
		→			

Column 1 System name

Column 2 Queue manager name

Column 3 Channel - showing direction of message flow.
If the arrow points right, this is a sender channel. If it points to the left, this is a receiver channel.

Column 4 System name of communicating system

Column 5 Communicating queue manager name

Column 6 Name assigned to the communication channel

Figure 17. Definition of the Channels. The channel names used in the connected systems do not have to be symmetrical.

A.6 Transact Configuration (Routing Table) - Worksheet

Each object must be defined to the queue manager.

There must be an entry in this worksheet for every local or remote queue in Figure 17. In addition there must be a transmission queue defined for every sender channel (those with right pointing arrows in Figure 17).

An explanation of whether entries are required, optional or not applicable is given in Figure 20. If aliases are required then they should be defined here. Aliases are more important in large networks, so they are strongly recommended if the network is large or is likely to grow. Because of the simplicity of our system we did not use aliases.

PRODCICS(SSIQM)

Obj_Name	Type	Queue_Name	Q_Mgr_Name	Xmit_Q_Name
F4L	Local			
	Local			
	Local			
	Local			
XMIT4	Transmit			
	Transmit			
F8R	Remote	F8L	SSIF8	XMIT4
	Remote			

Column 1 Object name - the name of the object.

Column 2 The object type, which can be a local, remote or transmission queue, or one of the three types of alias.

Column 3 Queue name. For a remote queue, or a remote alias, this is the name of the remote queue. For a queue alias, it is the name of the queue for which an alias is being provided.

Column 4 Queue manager name. For a remote queue, or a remote alias, this is the name of the manager of the remote queue. For a manager alias, it is the name of the manager for which an alias is being provided.

Column 5 Transmission Queue name. If this is not provided for a remote queue definition ezBRIDGE Transact for VSE will assume a default value equal to the queue manager name in column 4.

Figure 18. Definition of the Routing Table for PRODCICS. Required options are shown in Figure 20 on page 44.

CICSF8(SSIF8)

Obj_Name	Type	Queue_Name	Q_Mgr_Name	Xmit_Q_Name
F8L	Local			
	Local			
	Local			
	Local			
XMIT8	Transmit			
	Transmit			
F4R	Remote	F4L	SSIQM	XMIT8
	Remote			

Column 1 Object name - the name of the object.

Column 2 The object type, which can be a local, remote or transmission queue, or one of the three types of alias.

Column 3 Queue name. For a remote queue, or a remote alias, this is the name of the remote queue. For a queue alias, it is the name of the queue for which an alias is being provided.

Column 4 Queue manager name. For a remote queue, or a remote alias, this is the name of the manager of the remote queue. For a manager alias, it is the name of the manager for which an alias is being provided.

Column 5 Transmission Queue name. If this is not provided for a remote queue definition ezBRIDGE Transact for VSE will assume a default value equal to the queue manager name in column 4.

Figure 19. Definition of the Routing Table for CICSF8. Required options are shown in Figure 20 on page 44.

Entries are of the following formats:

Obj_Name	Type	Queue_Name	Q_Mgr_Name	Xmit_Q_Name
Required	Local	n/a	n/a	n/a
Required	Transmit	n/a	n/a	n/a
Required	Remote	Required	Required	Optional
Required	Alias_Q	Required	n/a	n/a
Required	Alias_M	n/a	Required	Optional
Required	Alias_R	Required	Required	n/a

Figure 20. Routing Table Legend. This shows whether a variable is required, optional or not applicable for the different object types.

Appendix B. Jobs Used to Install ezBRIDGE Transact on VSE/ESA

```
* $$ JOB JNM=RESTOR1,CLASS=0,DISP=D,NTFY=YES
// JOB  PLR1  DEFINE FILE
// EXEC IDCAMS,SIZE=AUTO
 DEFINE CLUSTER ( -
       NAME (VSE.MQMUSR1.LIBRARY                    ) -
       CYL (2 1) -
       SHAREOPTIONS (3) -
       RECORDFORMAT (NOCIFORMAT       ) -
       VOLUMES (DSWWRK ) -
       NOREUSE -
       NONINDEXED -
       TO (99366)) -
       DATA (NAME (VSE.MQMUSR1.LIBRARY.@D@                ) ) -
       CATALOG (EZBRIDGE.CATALOG                     )
  IF LASTCC NE 0 THEN CANCEL JOB
/*
// OPTION STDLABEL=ADD
// DLBL MQMUSR1,'VSE.MQMUSR1.LIBRARY',,VSAM,                    X
            CAT=MQMCAT,DISP=(OLD,KEEP)
/*
// EXEC IESVCLUP,SIZE=AUTO
A VSE.MQMUSR1.LIBRARY                    MQMUSR1 MQMCAT  OLD KEEP
/*
// EXEC LIBR,PARM='MSHP'
      DEFINE LIB=MQMUSR1   REPLACE=YES
/*
/&
* $$ EOJ
```

Figure 21. Job to Define the User Library MQMUSR1

Note that the control statements for IESVCLUP are highly positional, and the offsets for the elements are given in Table 1. The user catalog and enough space for the library must have been defined before you run this job.

Table 1. Offsets of the Control Statement for IESVCLUP

Character string	**Starting column**
A	1
VSE.MQMUSR1.LIBRARY	3
MQMUSR1	48
MQMCAT	56
OLD KEEP	64

```
* $$ JOB JNM=MQMLOAD,CLASS=8,DISP=D
// JOB MQMLOAD
// DLBL MQMCAT,'EZBRIDGE.CATALOG',,VSAM
// DLBL MQMUSR1,'VSE.MQMUSR1.LIBRARY',,VSAM,CAT=MQMCAT
// PAUSE LOAD THE EZBRIDGE TAPE ON 181
// MTC REW,181
// ASSGN SYS006,181
// EXEC LIBR
   RESTORE LIB=MQMUSR1 T=SYS006 LIST=Y R=Y
/*
/&
* $$ EOJ
```

Figure 22. Job to Restore the ezBRIDGE Distribution Tape

This job restores the distribution tape. It is the same as that in the ezBRIDGE Transact on VSE/ESA for MQSeries User's Guide, except that minor typographical errors have been corrected.

```
* $$ JOB JNM=DFHPCTEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHPCTC2 ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
*****************************************************
*                                                   *
*  5686-028 (C) COPYRIGHT IBM CORP. 1984, 1990      *
*                                                   *
*****************************************************
       TITLE 'DFHPCTEZ -- EZBRIDGE xacts'
       PUNCH ' CATALOG DFHPCTEZ.OBJ REP=YES'
       DFHPCT TYPE=INITIAL,SUFFIX=EZ
*-------------------------------------------------------------------*
       SPACE 3
       SPACE 3
*-------------------------------------------------------------------*
*        LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX              *
*-------------------------------------------------------------------*
       SPACE 3
* $$ SLI MEM=CICSPCT.USER,S=MQMUSR1.USER
*-------------------------------------------------------------------*
*        LOCAL ENTRIES SHOULD BE PLACED ABOVE THIS BOX              *
*-------------------------------------------------------------------*
       SPACE 3
       DFHPCT TYPE=FINAL
       END   DFHPCTBA
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 23. Job to Define the PCT Entries for ezBRIDGE

This job will build a table with just the ezBRIDGE Transact on VSE/ESA for MQSeries transactions, plus any required CICS transactions. It is intended that the generated table should be migrated to the CSD. When this is done, only the ezBRIDGE transactions will actually be migrated.

```
* $$ JOB JNM=DFHPPTEZ,CLASS=A,DISP=D,NTFY=YES
* $$ LST CLASS=Q,DISP=H
// JOB DFHPPTC2 ASSEMBLE
// LIBDEF *,CATALOG=PRD2.CONFIG
// OPTION CATAL,LIST
// EXEC ASSEMBLY
*************************************************
*                                               *
*  5686-028 (C) COPYRIGHT IBM CORP. 1984, 1990  *
*                                               *
*************************************************
      TITLE 'DFHPPTEZ -- ezBRIDGE programs'
      PUNCH ' CATALOG DFHPPTEZ.OBJ REP=YES'
      DFHPPT TYPE=INITIAL,SUFFIX=EZ
      SPACE 3
*---------------------------------------------------------------*
*       LOCAL ENTRIES SHOULD BE PLACED BELOW THIS BOX           *
*---------------------------------------------------------------*
* $$ SLI MEM=CICSPPT.USER,S=MQMUSR1.USER
      SPACE 3
*---------------------------------------------------------------*
*       LOCAL ENTRIES SHOULD BE PLACED ABOVE THIS BOX           *
*---------------------------------------------------------------*
      SPACE 3
      DFHPPT TYPE=FINAL
      END   DFHPPTBA
/*
// EXEC LNKEDT
/*
/&
* $$ EOJ
```

Figure 24. Job to Define the PPT Entries for ezBRIDGE

This job will build a table with just the ezBRIDGE Transact on VSE/ESA for MQSeries programs. It is intended that the generated table should be migrated to the CSD.

```
// JOB CSDJOB   JOB TO EXECUTE DFHCSDUP
// DLBL DFHCSD,'CICS.CSD',,VSAM,CAT=VSESPUC
// LIBDEF PHASE,SEARCH=PRD2.CONFIG
*
*
// EXEC DFHCSDUP,SIZE=300K
  MIGRATE TABLE(DFHPCTEZ) TOGROUP(EZPCT2)
  MIGRATE TABLE(DFHPPTEZ) TOGROUP(EZPPT2)
/*
/&
```

Figure 25. Job to Migrate the Table Definitions for ezBRIDGE

This job migrates the program and transaction definitions to the CSD, and puts them into the same group. This group should be added to the start list of the system or systems where ezBRIDGE on VSE is to be used.

Appendix C. APPC Definitions Used to Communicate Between PRODCICS and CICSF8

Sample definitions for both CICS systems and for VTAM are shown. Note that we held both CICS systems in the same VSE, and consequently there was no need for cross-domain definitions, or indeed for NCP changes.

C.1 CICS Definitions

The definitions for each CICS system were placed in a single group, which was manually installed. This group, of course, would normally be added to the start list. The CSD was shared between CICS systems for operational ease.

```
CEDA EXP GROUP(CICSF4)
ENTER COMMANDS
 NAME     TYPE         GROUP                                   DATE   TIME
 C4T8     CONNECTION   CICSF4                                  94.041 14.32.03
 CICS4    SESSIONS     CICSF4                                  94.059 11.58.08
```

Figure 26. Group CICSF4. This group contains the definitions to be used in PRODCICS for the connection and sessions to CICSF8.

```
OBJECT CHARACTERISTICS
 CEDA  View
  Connection     : C4T8
  Group          : CICSF4
 CONNECTION IDENTIFIERS
  Netname        : CICSF8
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam              Vtam | IRc | INdirect
  Protocol       : Appc              Appc | Lu61
  SInglesess     : No                No | Yes
  Datastream     : User              User | 3270 | SCs | STrfield | Lms
  RECordformat   : U                 U | Vb
 OPERATIONAL PROPERTIES
  AUtoconnect    : No                No | Yes | All
  INService      : Yes               Yes | No
  SECURITY
   SEcurityname   :
   ATtachsec      : Local              Local | Identify | Verify
   Bindpassword   :                    PASSWORD NOT SPECIFIED
```

Figure 27. Connection C4T8. This definition is used in PRODCICS for the connection to CICSF8.

The connection name is the value used in the Connection ID fields in the channel definitions for SSIQM, the queue manager in PRODCICS.

```
OBJECT CHARACTERISTICS
 CEDA  View
  Sessions       : CICS4
  Group          : CICSF4
 SESSION IDENTIFIERS
  Connection     : C4T8
  SESSName       :
  NETnameq       :
  MOdename       :
 SESSION PROPERTIES
  Protocol       : Appc               Appc | Lu61
  MAximum        : 00050 , 00020      0-32767
  RECEIVEPfx     :
  RECEIVECount   : No                 No | 1-999
  SENDPfx        :
  SENDCount      : No                 No | 1-999
  SENDSize       : 04096              1-30720
  RECEIVESize    : 04096              1-30720
 OPERATOR DEFAULTS
  OPERId         :
  OPERPriority   : 000                0-255
  OPERRsl        : 0                                                  0-24,.
  OPERSecurity   : 1                                                  1-64,.
  USERId         :
 SESSION USAGES
  Transaction    :
  SESSPriority   : 000                0-255
 OPERATIONAL PROPERTIES
  Autoconnect    : Yes                No | Yes | All
  INservice      :                    No | Yes
  Buildchain     : Yes                Yes | No
  USERArealen    : 000                0-255
  IOarealen      : 00000 , 00000      0-32767
  RELreq         : No                 No | Yes
  Discreq        : No                 No | Yes
  NEPclass       : 000                0-255
 RECOVERY
  RECOvoption    : Sysdefault         Sysdefault | None
```

Figure 28. Sessions CICS4. These sessions are used in PRODCICS for the link to CICSF8.

This defines the sessions on connection C4T8. Both maximum values in Session Properties have been increased as the default was too low. The values were deliberately set higher than anticipated, so that a reasonable setting could be obtained from the statistics.

```
CEDA EXP GROUP(CICSF8)
ENTER COMMANDS
 NAME     TYPE         GROUP                                  DATE   TIME
 C8T4     CONNECTION   CICSF8                                 94.041 16.13.08
 CICS8    SESSIONS     CICSF8                                 94.059 11.59.27
```

Figure 29. Group CICSF8. This group contains the definitions to be used in CICSF8 for the connection and sessions to PRODCICS.

```
OBJECT CHARACTERISTICS
 CEDA  View
  Connection     : C8T4
  Group          : CICSF8
 CONNECTION IDENTIFIERS
  Netname        : PRODCICS
  INDsys         :
 REMOTE ATTRIBUTES
  REMOTESystem   :
  REMOTEName     :
 CONNECTION PROPERTIES
  ACcessmethod   : Vtam              Vtam | IRc | INdirect
  Protocol       : Appc              Appc | Lu61
  SInglesess     : No                No | Yes
  Datastream     : User              User | 3270 | SCs | STrfield | Lms
  RECordformat   : U                 U | Vb
 OPERATIONAL PROPERTIES
  AUtoconnect    : No                No | Yes | All
  INService      : Yes               Yes | No
 SECURITY
  SEcurityname   :
  ATtachsec      : Local             Local | Identify | Verify
  Bindpassword   :                   PASSWORD NOT SPECIFIED
```

Figure 30. Connection C8T4. This definition is used in CICSF8 for the connection to PRODCICS.

The connection name is the value used in the Connection ID fields in the channel definitions for SSIF8, the queue manager in CICSF8.

```
OBJECT CHARACTERISTICS
 CEDA  View
  Sessions       : CICS8
  Group          : CICSF8
 SESSION IDENTIFIERS
  Connection     : C8T4
  SESSName       :
  NETnameq       :
  MOdename       :
 SESSION PROPERTIES
  Protocol       : Appc               Appc | Lu61
  MAximum        : 00050 , 00020      0-32767
  RECEIVEPfx     :
  RECEIVECount   : No                 No | 1-999
  SENDPfx        :
  SENDCount      : No                 No | 1-999
  SENDSize       : 04096              1-30720
  RECEIVESize    : 04096              1-30720
 OPERATOR DEFAULTS
  OPERId         :
  OPERPriority   : 000                0-255
  OPERRsl        : 0                                                  0-24,..
  OPERSecurity   : 1                                                  1-64,..
  USERId         :
 SESSION USAGES
  Transaction    :
  SESSPriority   : 000                0-255
 OPERATIONAL PROPERTIES
  Autoconnect    : Yes                No | Yes | All
  INservice      :                    No | Yes
  Buildchain     : Yes                Yes | No
  USERArealen    : 000                0-255
  IOarealen      : 00000 , 00000      0-32767
  RELreq         : No                 No | Yes
  Discreq        : No                 No | Yes
  NEPclass       : 000                0-255
 RECOVERY
  RECOvoption    : Sysdefault         Sysdefault | None
```

Figure 31. Sessions CICS8. These are the session definitions to be used in CICSF8 for the sessions to PRODCICS.

This defines the sessions on connection C8T4. Both maximum values in Session Properties have been increased as the default was too low. The values were deliberately set higher than anticipated, so that a reasonable setting could be obtained from the statistics.

C.2 VTAM Definitions

Since both VTAM applications were in the same VSE only one VTAM definition book was needed.

```
VTMAPPL  VBUILD TYPE=APPL
DBDCCICS APPL  AUTH=(PASS,ACQ,VPACE),PARSESS=YES,EAS=15,VPACING=3
PRODCICS APPL  AUTH=(PASS,ACQ,VPACE),PARSESS=YES,EAS=15,VPACING=3
CICSF8   APPL  AUTH=(PASS,ACQ,VPACE),PARSESS=YES,EAS=15,VPACING=3
POWER    APPL  AUTH=(ACQ)
PNET     APPL  AUTH=(PASS,ACQ),VPACING=3,MODETAB=VTMLOGTB,DLOGMOD=PNET
IESWAITT APPL  AUTH=(NOACQ)
```

Figure 32. VTAM Application Changes. The changes made are highlighted.

Appendix D. Definitions Used for Distributed Queue Processing

```
Queue definition - transmit
03/07/94            ezBRIDGE Transact for the MQSeries              PRODCICS
15:16:10                Queue Definition  Record                    CIC2
MQFCNFG           QM - SSIQM                                        A002

                      LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : XMIT4
 Description line 1 . . . . : SEND TO F8
 Description line 2 . . . . :

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No
Usage mode . . . . . . . . . : T     N=Normal, T=Transmission
Share mode . . . . . . . . . : Y     Y=Yes, N=No
Physical File name . . . . : MQFI001    MQSERIES.MQFI001
                           Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00003353     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                         Trigger Information
Trigger enable . . . . . . . : Y     Y=yes, N=No
Trigger Type   . . . . . . . : F     F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID : A001     SYSID      :
Program ID : MQPSEND                      Remote CID : CH428

RECORD DISPLAYED.
```

Figure 33. Definition of the Transmission Queue in PRODCICS (F4). What is shown here is a concatenation of the basic Local Queue Definition record and the Local Queue Extended Definition.

Since this is a transmission queue for which triggering is required, the following fixed settings are needed.

Trigger enable	Y
Trigger type	F
Maximum Trigger Starts	0001
Program ID	MQPSEND
Remote CID	CH428 (the channel to be used)
Trans ID	**must** be blank

```
03/07/94           ezBRIDGE Transact for the MQSeries              PRODCICS
15:17:28               Queue Definition  Record                    CIC2
MQFCNFG          QM - SSIQM                                        AOO2

                      REMOTE QUEUE DEFINITION

 Object Name. . . . . . . . : F8R
 Description line 1 . . . . : REMOTE Q IN F8 TO RECEIVE OUTPUT
 Description line 2 . . . . : REQUESTS

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No

 Remote QUEUE name . . . . .  F8L
 Remote QM    name . . . . .  SSIF8

 Transmission name . . . . .  XMIT4
```

Figure 34. Definition of the Remote Queue in PRODCICS (F4)

The values used in this object are those documented in Figure 18 on page 42.

```
03/07/94          ezBRIDGE Transact for the MQSeries                PRODCICS
15:15:12                    Channel Record          DISPLAY         CIC2
           Last check point               Last update  19940307    AOO2
MSN 00000001  Time 15:06:48  Interv 000000  Create date  19940221

 Channel name  : CH428                       Channel type    S S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

       Allocation retries               Get retries
   Number of retries: 00000005       Number of retries:  00000002
   Delay time-fast :  00000002        Delay time      :  00000005
   Delay time-slow :  00000005

  Max messages per batch : 000001         Max transmission size  :  003353
  Message sequence wrap  : 000001         Max message size       :  003353

   Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
             Connection ID: C4T8          Remote task ID: MQ01
 Transmit queue name        XMIT4

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) N      Dead letter store(Y/N) N
```

Figure 35. Definition of the Channel in PRODCICS (F4)

The values for this definition are taken from the line for this object in Figure 17 on page 41. The channel type is S because this is a sender channel (indicated in Figure 17 by the right arrow). The protocol is L62 (fixed) and the format is fixed as MCP. The Connection ID is the name of the CICS connection which is to be used, and the associated transmission queue is XMIT4. Since the channel terminates in another CICS/VSE system, the Remote Task ID must be MQ01, which is the CICS/VSE transaction associated with the channel receiver program.

```
MQIC75 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:48
CHAN: CH428                QUEU:
MSN:00000000**INFORMATION: CHANNEL CONNECT  *RC:00
MQIC78 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:48
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: LU62 CONNECT     *RC:00
MQIC76 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:48
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: CHANNEL OPENED   *RC:00
MQIMO2 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:48
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: NEGOTIATION COMP *RC:00
NEG PARMS: MAX-MSG-PER-BATCH 000001 MAX-TRANS-SIZE 003353 MAX-MSG-SIZE 003353
     MAX-SEQ-WRAP 000001 FLAG-SEQ-NO Y FLAG-CONV-CAPABLE N FLAG-SPLIT-MESS N
MQIC80 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:54
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: QUEUE DEPLETED   *RC:00
MQIC83 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:54
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: CHANNEL SHUT SENT*RC:00
MQIC82 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:54
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: CHANNEL DISC     *RC:00
MQIC83 PRG: MQPSEND  TRN: MQO2 TSK:00787 03/07/94 15:06:54
CHAN: CH428                QUEU: XMIT4
MSN:00000001**INFORMATION: CHANNEL SHUT     *RC:00
```

Figure 36. Message Log Listing for PRODCICS (F4). This log was created during a successful transfer of data over the link. The messages were produced on CSMT because no LOG queue was defined.

Notes:

1. This figure is the sender for which Figure 39 on page 57 is the receiver
2. On the first request the channel only is identified
3. On the second request the channel and transmitter queue are both identified
4. However many messages are read to deplete the transmission queue, MQIC80 only appears once
5. Since this is the channel sender, the program identified is MQPSEND
6. Return codes are consistently zeroes

```
Channel definition
03/07/94          ezBRIDGE Transact for the MQSeries                CICSF8
15:21:51                    Channel Record           DISPLAY         CIC8
            Last check point                Last update  19940307    A002
MSN 00000001  Time 15:06:48  Interv 000000  Create date  19940301

 Channel name  : CH428                       Channel type    R S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

       Allocation retries                Get retries
   Number of retries: 00000005        Number of retries:  00000002
   Delay time-fast :  00000002        Delay time      :   00000005
   Delay time-slow :  00000005

 Max messages per batch : 000001         Max transmission size  :  003353
 Message sequence wrap  : 000001         Max message size       :  003353

  Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
            Connection ID:               Remote task ID:
 Transmit queue name

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) N      Dead letter store(Y/N) N
```

Figure 37. Channel Definition for CICSF8

The values for this definition are taken from the line for this object in Figure 17 on page 41. The channel type is R because this is a receiver channel (indicated in Figure 17 by the left arrow). The protocol is L62 (fixed) and the format is fixed as MCP. The Connection Id is the name of the CICS connection which is to be used. Since this is a receiver channel you should not specify any values for the Connection ID, Remote task ID or Transmit queue name.

```
03/07/94            ezBRIDGE Transact for the MQSeries              CICSF8
15:22:38                Queue Definition  Record                    CIC8
MQFCNFG           QM - SSIF8                                        A002

                      LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : F8L
 Description line 1 . . . . : RECEIVES FROM F4
 Description line 2 . . . . :

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No


Usage mode . . . . . . . . . : N     N=Normal, T=Transmission
Share mode . . . . . . . . . : N     Y=Yes, N=No
Physical File name . . . . : MQFI002    MQSERIES.MQFI0028
                               Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00003353     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                           Trigger Information
Trigger enable . . . . . . . : N     Y=yes, N=No
Trigger Type   . . . . . . . :       F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID :          SYSID      :
Program ID :                              Remote CID :
```

Figure 38. Local Queue Definition for CICSF8. As well as being the local queue for SSIF8, this queue is also defined as remote to SSIQM. Although this is referred to by a remote queue definition in another system, it is simply a local queue definition, and corresponds to the first line in Figure 19.

```
MQIC75 PRG: MQPRECV  TRN: MQ01 TSK:00580 03/07/94 15:06:48
CHAN: CH428                QUEU:
MSN:00000000**INFORMATION: CHANNEL CONNECT  *RC:00
MQIM02 PRG: MQPRECV  TRN: MQ01 TSK:00580 03/07/94 15:06:48
CHAN: CH428                QUEU:
MSN:00000001**INFORMATION: NEGOTIATION COMP *RC:00
NEG PARMS: MAX-MSG-PER-BATCH 000001 MAX-TRANS-SIZE 003353 MAX-MSG-SIZE 003353
     MAX-SEQ-WRAP 000001 FLAG-SEQ-NO Y FLAG-CONV-CAPABLE N FLAG-SPLIT-MESS N
MQIC77 PRG: MQPRECV  TRN: MQ01 TSK:00580 03/07/94 15:06:48
CHAN: CH428                QUEU: F8L
MSN:00000001**INFORMATION: QUEUE OPENED     *RC:00
MQIC40 PRG: MQPRECV  TRN: MQ01 TSK:00580 03/07/94 15:06:54
CHAN: CH428                QUEU: F8L
MSN:00000001**CHANNEL DOWN                  *RC:00
MQIC82 PRG: MQPRECV  TRN: MQ01 TSK:00580 03/07/94 15:06:54
CHAN: CH428                QUEU: F8L
MSN:00000001**INFORMATION: CHANNEL DISC     *RC:00
```

Figure 39. Message Log Output in CICSF8

This was obtained by accessing the View Message Log dialog of the Interactive Interface. It shows the messages written to the destination CSMT by a successful transfer of data from the application WRTOF8 into the queue F8L at SSIF8.

```
03/15/94           ezBRIDGE Transact for the MQSeries              PRODCICS
10:56:12               Queue Definition  Record                    CIC2
MQFCNFG          QM - SSIQM                                        A002

                      LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : F4L
 Description line 1 . . . . : LOCAL Q FOR F4
 Description line 2 . . . . :

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No
                        Local Queue Information
Usage mode . . . . . . . . : N     N=Normal, T=Transmission
Share mode . . . . . . . . : Y     Y=Yes, N=No
Physical File name . . . . : MQFI001    MQSERIES.MQFI001
                             Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00003353     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                           Trigger Information
Trigger enable . . . . . . : N     Y=yes, N=No
Trigger Type   . . . . . . :       F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID :          SYSID      :
Program ID :                              Remote CID :
```

Figure 40. Definition in PRODCICS of the Local Receiver Queue

This corresponds to the first line in Figure 18.

```
03/15/94          ezBRIDGE Transact for the MQSeries                PRODCICS
10:57:26                    Channel Record          DISPLAY         CIC2
           Last check point                Last update  19940309    A002
MSN 00000001  Time 14:19:02  Interv 000000  Create date  19940309

 Channel name  : CH824                       Channel type    R S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

       Allocation retries                Get retries
   Number of retries: 00000005        Number of retries:  00000002
   Delay time-fast :  00000002        Delay time      :   00000005
   Delay time-slow :  00000005

  Max messages per batch : 000001         Max transmission size  :  003353
  Message sequence wrap  : 000001         Max message size       :  003353

   Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
             Connection ID:               Remote task ID:
 Transmit queue name

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) N      Dead letter store(Y/N) N
```

Figure 41. Definition in PRODCICS of the Receiver Channel

This corresponds to the first line in Figure 17.

```
03/15/94            ezBRIDGE Transact for the MQSeries              CICSF8
10:58:34                Queue Definition  Record                    CIC8
MQFCNFG           QM - SSIF8                                        A002

                      REMOTE QUEUE DEFINITION

 Object Name. . . . . . . . : F4R
 Description line 1 . . . . : REMOTE Q IN F4 TO RECEIVE OUTPUT
 Description line 2 . . . . : REQUESTS

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No

 Remote QUEUE name . . . . .  F4L
 Remote QM    name . . . . .  SSIQM

 Transmission name . . . . .  XMIT8
```

Figure 42. Definition in CICSF8 of the Remote Queue which is F4L in PRODCICS

This corresponds to the seventh line in Figure 18.

```
03/15/94            ezBRIDGE Transact for the MQSeries              CICSF8
10:59:20                Queue Definition  Record                    CIC8
MQFCNFG           QM - SSIF8                                        A002

                       LOCAL QUEUE DEFINITION

 Object Name. . . . . . . . : XMIT8
 Description line 1 . . . . : SEND TO F4
 Description line 2 . . . . :

 Put Enabled  . . . . . . . : Y      Y=Yes, N=No
 Get Enabled  . . . . . . . : Y      Y=Yes, N=No
                     Local Queue Information
Usage mode . . . . . . . . : T     N=Normal, T=Transmission
Share mode . . . . . . . . : Y     Y=Yes, N=No
Physical File name . . . . : MQFI001    DSN EXIST. FILE IS CLOSED.
                        Maximum Values
Maximum Q Depth. . . . . . : 00100000     Global Lock entries . : 00000100
Maximum message length . . : 00003353     Local Lock entries. . : 00000100
Maximum number of Opens . .  00000100     Checkpoint threshold  : 1000

                       Trigger Information
Trigger enable . . . . . . : Y     Y=yes, N=No
Trigger Type   . . . . . . : F     F=First, E=every
Maximum Trigger starts . . : 0001
Trans ID   :           Term ID : A001     SYSID      :
Program ID : MQPSEND                      Remote CID : CH824
```

Figure 43. Definition in CICSF8 of the Transmission Queue to PRODCICS

Since this is a transmission queue for which triggering is required, the following fixed settings are needed.

Trigger enable Y
Trigger type F
Maximum Trigger Starts 0001
Program ID MQPSEND

Remote CID CH824 (the channel to be used)
Trans ID **must** be blank

This corresponds to the seventh line in Figure 18.

```
03/15/94          ezBRIDGE Transact for the MQSeries                CICSF8
11:00:34                    Channel Record          DISPLAY         CIC8
           Last check point                Last update  19940310    A002
MSN 00000001  Time 11:00:06  Interv 000000  Create date  19940309

 Channel name  : CH824                       Channel type    S S=Send/R=Recv
 Protocol    L62    L62/BSC/LAN/X25          Format   MCP    MLP/MEP/MCP

       Allocation retries                Get retries
   Number of retries: 00000005        Number of retries:  00000002
   Delay time-fast :  00000002        Delay time      :   00000005
   Delay time-slow :  00000005

  Max messages per batch : 000001         Max transmission size  :  003353
  Message sequence wrap  : 000001         Max message size       :  003353

   Mess seq req(y/n): Y     Convers cap (y/n): N      Split mess(y/n):   N
             Connection ID: C8T4          Remote task ID: MQ01
 Transmit queue name        XMIT8

 Checkpoint values:       Frequency   0020     Time span   0010
 Enable(Y/N) N      Dead letter store(Y/N) N
```

Figure 44. Definition in CICSF8 of the Sender Channel to PRODCICS

This corresponds to the second line in Figure 18.

Appendix E. Components for Compiling and Running the Interface Sample

The sample code given in the following appendix is the code used for the batch implementation. The application program (CALLER) is based on the sample program TTPTST2, provided as part of ezBRIDGE Transact on VSE/ESA for MQSeries TTPTST2 is copyrighted, and is the property of Systems Strategies Inc. Their permission was asked and obtained for including the modified copy in this book. The objective in basing our batch sample on this was that the changes made should be visible, and easily verified against a working application.

The online COBOL program is designed to translate the requests received in a COMMAREA to a set of individual calls conforming to the MQI.

E.1 A Method of Capturing Source Code from Softcopy Documentation

The following method can be used to capture source code from softcopy using BookManager Read/2. A variation of this can be used if the softcopy is accessed under VM. This method should only be used to capture those programs which are not protected by copyright (that is EZBMTASK, EZBMBTCH, MQICICS, STOPXPCC and TASKCAN). It may also be used to copy job-streams to be used in installation of ezBRIDGE Transact on VSE/ESA for MQSeries and during the activation of the batch interface.

1. Open the book
2. With the table of contents in foreground, select ′OPTIONS′
3. Within options, select ′Tree′
4. Select ′Expand all headings′
5. Click once on the component required on the table of contents
6. Select ′SERVICES′
7. Select ′Copy to Clipboard′

   All available components will fit in the 64K available in the clipboard
8. Invoke your favorite PS/2* editor for a new file
9. Paste from the clipboard
10. Upload the file to the VSE Host Transfer File
11. Move the file to ICCF

E.2 Batch COBOL Program (CALLER) Showing the Sample Interface

This program is in a consecutive piece. It will not fit into the OS/2 clipboard. Since it is copyright SSI, it should not be copied, although the techniques contained herein may be used.

E.2.1 Source Code for the Batch Application Sample

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    CALLER.
 AUTHOR.        IBM ITSO CENTER BOEBLINGEN.
*-----------------------------------------------------------------*
* COPYRIGHT :                                                     *
*                                                                 *
* THIS PROGRAM IS BASED (WITH PERMISSION OF SYSTEMS STRATEGIES    *
* INC.) ON A SAMPLE PROGRAM WHICH IS THE                          *
* PROPRIETARY INFORMATION OF SYSTEMS STRATEGIES INC.              *
* COPYRIGHT (C) AS AN UNPUBLISHED WORK IN 1988 TO 1992;           *
* ALL RIGHTS RESERVED.                                            *
*                                                                 *
* COPYING, ADAPTATION, DISPLAY, USE OR DISCLOSURE WITHOUT         *
* WRITTEN CONSENT OF SYSTEMS STRATEGIES INC IS PROHIBITED.        *
*-----------------------------------------------------------------*
*****************************************************************
* PROGRAM NAME : CALLER - CALLING PROGRAM FOR THE EZBRIDGE BATCH *
*                EXTENSIONS INTERFACE PROGRAM                     *
*                                                                 *
* CALLS        : EZBMBTCH - THE SUBROUTINE TO DO THE XPCC CALLS   *
*                TO THE CICS SUBTASK.                             *
*                                                                 *
*****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  FILLER                          PIC X(40) VALUE
                  'CALLER:- WORKING STORAGE STARTS HERE =>'.
 77  MQCLOSE                         PIC X(08) VALUE 'MQCLOSE'.
 77  MQCONN                          PIC X(08) VALUE 'MQCONN '.
 77  MQDISC                          PIC X(08) VALUE 'MQDISC '.
 77  MQGET                           PIC X(08) VALUE 'MQGET  '.
 77  MQINQ                           PIC X(08) VALUE 'MQINQ  '.
 77  MQOPEN                          PIC X(08) VALUE 'MQOPEN '.
 77  MQPUT                           PIC X(08) VALUE 'MQPUT  '.
 77  MQPUT1                          PIC X(08) VALUE 'MQPUT1 '.
 77  EZDISC                          PIC X(08) VALUE 'EZDISC '.
 77  WORK1                           PIC 9(08).
 77  WORK2                           PIC 9(08).
 01  WS-WORK-FIELDS.
     05  WS-IDX                      PIC S9(4)  COMP VALUE ZERO.
     05  WS-COUNT                    PIC S9(4)  COMP VALUE ZERO.
     05  WS-PROCESS-TIMES            PIC  9(4)       VALUE ZERO.
     05  WS-DURATION-SECS            PIC  X(8)       VALUE SPACES.
     05  WS-PASS-MSG-LENGTH          PIC S9(4)  COMP   VALUE ZERO.
     05  WS-APPL-MSG-LENGTH          PIC S9(8)  COMP   VALUE ZERO.
     05  WS-ABSTIME                  PIC S9(15) COMP-3 VALUE ZERO.
     05  WS-ABSTIME2                 PIC S9(15) COMP-3 VALUE ZERO.
     05  WS-DATE.
        10  WS-DATE-CC               PIC  99  VALUE ZERO.
        10  WS-DATE-YYMMDD.
          12  WS-DATE-YY             PIC  99  VALUE ZERO.
          12  WS-DATE-MM             PIC  99  VALUE ZERO.
          12  WS-DATE-DD             PIC  99  VALUE ZERO.
     05  WS-TIME-9                   PIC  9(7)  VALUE ZERO.
     05  WS-TIME  REDEFINES  WS-TIME-9.
        10  FILLER                   PIC  9.
```

```
        10  WS-TIME-HHMMSS.
          12  WS-TIME-HH            PIC  99.
          12  WS-TIME-MM            PIC  99.
          12  WS-TIME-SS            PIC  99.
      05  WS-FORMATTED-TIME.
          10  WS-FORMAT-TIME-HH     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE ':'.
          10  WS-FORMAT-TIME-MM     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE ':'.
          10  WS-FORMAT-TIME-SS     PIC  X(02) VALUE SPACES.
      05  WS-FORMATTED-DATE.
          10  WS-FORMAT-DATE-MM     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE '/'.
          10  WS-FORMAT-DATE-DD     PIC  X(02) VALUE SPACES.
          10  FILLER                PIC  X(01) VALUE '/'.
          10  WS-FORMAT-DATE-YY     PIC  X(02) VALUE SPACES.
      05  WS-QM-Q-NAME.
        10  WS-QM-NAME              PIC  X(48) VALUE 'QM1   '.
        10  WS-Q-NAME               PIC  X(48) VALUE 'QUEUE'.
      05  WS-REPLY-Q                PIC  X(48) VALUE 'QUE1'.
      05  WS-ERR-MSG-FLAG           PIC  X     VALUE SPACES.
       88 WS-ERR-MSG                          VALUE 'Y'.
      05  WS-STARTED-FLAG           PIC  X     VALUE SPACES.
       88 WS-STARTED                          VALUE 'Y'.
      05  WS-TIMESTAMP              PIC  X     VALUE SPACES.
       88 WS-PUT-TIMESTAMP                    VALUE 'Y'.
      05  WS-TIMESTAMP-VALUE.
         10  WS-TIMESTAMP-DATE      PIC X(6) VALUE SPACES.
         10  WS-TIMESTAMP-TIME      PIC X(6) VALUE SPACES.
      05  WS-END-OF-MESSAGES-FLAG   PIC  X  VALUE SPACES.
        88 WS-END-OF-MESSAGES              VALUE 'Y'.
      05  WS-TRUNCATED-MESSAGES-F   PIC  X  VALUE SPACES.
        88 WS-TRUNCATED-MESSAGES           VALUE 'Y'.
*-----------------------------------------------------------------*
  EJECT
*-----------------------------------------------------------------*
* ERROR PROCESSING - MESSAGE DEFINITIONS - MQIERRWS
*-----------------------------------------------------------------*
*--ERROR MESSAGE DEFINITION---------------------------------*
 01  ERR-MESSAGE.
     05 ERR-MSG-1.
        10 FILLER               PIC X(3) VALUE 'MQI'.
        10 ERROR-CODE           PIC X(3) VALUE SPACES.
*--INFORMATIONAL MESSAGES
          88 ERROR-INFO-OK               VALUE 'M01'.
          88 ERROR-INFO-OK-WITH-MSG      VALUE 'M02'.
*--PROCESSING ERRORS
          88 ERROR-PROC-QM-STOPPED       VALUE 'M10'.
          88 ERROR-PROC-Q-NAME-MISSING   VALUE 'M11'.
          88 ERROR-PROC-Q-DEPTH-EXCEED   VALUE 'M12'.
          88 ERROR-PROC-CONCURRENT-UPD   VALUE 'M13'.
          88 ERROR-PROC-NOT-AUTHORIZED   VALUE 'M14'.
          88 ERROR-PROC-SYSTEM-ACTIVE    VALUE 'M15'.
*--LOGIC ERRORS
          88 ERROR-LOGIC-NOT-SUPPORTED   VALUE 'M30'.
          88 ERROR-LOGIC-STARTED-WRONG   VALUE 'M31'.
          88 ERROR-LOGIC-MISSING-RECORD  VALUE 'M32'.
          88 ERROR-LOGIC-LOCKS-EXCEEDED  VALUE 'M33'.
          88 ERROR-LOGIC-REPEATED-FAILURE VALUE 'M35'.
```

```
          88 ERROR-LOGIC-RECORD-NOTFOUND  VALUE 'M36'.
          88 ERROR-LOGIC-RECORD-DUPLICATE VALUE 'M37'.
*--INTERNAL ERROR
          88 ERROR-INTERNAL-CALL-ERROR    VALUE 'M40'.
          88 ERROR-INTERNAL-STRUCTURE     VALUE 'M41'.
          88 ERROR-INTERNAL-MAPFAIL       VALUE 'M42'.
*--CICS TABLE/ENTRY ERROR
          88 ERROR-CICS-TABLE-MISSING     VALUE 'M80'.
          88 ERROR-CICS-NO-SPACES         VALUE 'M81'.
          88 ERROR-CICS-NO-STORAGE        VALUE 'M82'.
          88 ERROR-CICS-FILE-NOTOPEN      VALUE 'M83'.
          88 ERROR-CICS-DISABLED          VALUE 'M84'.
          88 ERROR-CICS-IO-ERROR          VALUE 'M85'.
          88 ERROR-CICS-QIDERROR          VALUE 'M86'.
          88 ERROR-CICS-ITEMERROR         VALUE 'M87'.
          88 ERROR-CICS-LENGERROR         VALUE 'M88'.
*--CICS MAJOR ERRORS
          88 ERROR-CICS-ERROR             VALUE 'M90'.
          88 ERROR-CICS-INVALID           VALUE 'M91'.
          88 ERROR-CICS-ILLOGIC           VALUE 'M92'.
          88 ERROR-CICS-ABEND             VALUE 'M99'.
*--COMMUNICATION ERRORS----------------
*---------------------- SYNCH ERRORS
          88  SYNCH-MSN-ERROR             VALUE 'C01'.
*---------------------- LU6.2 ERRORS
          88  LU62-FREE-ERROR             VALUE 'C10'.
          88  LU62-EIB-ERROR              VALUE 'C11'.
          88  LU62-STAT-ERROR             VALUE 'C12'.
          88  LU62-ALLOC-ERROR            VALUE 'C13'.
          88  LU62-ALLOC-RETRY-ERROR      VALUE 'C14'.
          88  LU62-CONN-ERROR             VALUE 'C15'.
          88  LU62-SEND-ERROR             VALUE 'C16'.
          88  LU62-RECV-RESP-ERROR        VALUE 'C17'.
          88  LU62-RECV-FREE-ERROR        VALUE 'C18'.
*---------------------- RESPONSE ERRORS
          88  INVLD-RESP-TYPE              VALUE 'C23'.
          88  INVLD-RESP-MSN               VALUE 'C24'.
          88  FATAL-RESP-TYPE              VALUE 'C25'.
*---------------------- PARSER ERRORS
          88  PARSER-MSG-ERROR             VALUE 'C29'.
          88  PARSER-BIG-ENDIAN-ERROR      VALUE 'C30'.
          88  PARSER-TSH-ERROR             VALUE 'C31'.
          88  PARSER-CCSID-ERROR           VALUE 'C32'.
          88  PARSER-MSH-ERROR             VALUE 'C33'.
          88  PARSER-MQX-ERROR             VALUE 'C34'.
          88  PARSER-INITIAL-ERROR         VALUE 'C35'.
          88  PARSER-FAP-ERROR             VALUE 'C36'.
          88  PARSER-MSG-SIZE-ERROR        VALUE 'C37'.
          88  PARSER-WRAP-ERROR            VALUE 'C38'.
          88  PARSER-MCP-DOWN-ERROR        VALUE 'C39'.
          88  PARSER-CHANNEL-DOWN          VALUE 'C40'.
          88  PARSER-CHANNEL-NOT-FOUND     VALUE 'C41'.
          88  PARSER-CHANNEL-ERROR         VALUE 'C42'.
          88  PARSER-CHANNEL-BUSY          VALUE 'C43'.
          88  PARSER-RESYNC-ERROR          VALUE 'C45'.
*         88  PARSER-STATUS-ERROR          VALUE 'C46'.
          88  PARSER-LENGTH-ERROR          VALUE 'C47'.
          88  PARSER-MAX-MSG-PER-BATCH     VALUE 'C48'.
          88  PARSER-MAX-TRANSM-SIZE       VALUE 'C49'.
```

```
*---------------------- QUEUER ERRORS
          88  MCCONN-ERROR                VALUE 'C51'.
          88  MQOPEN-ERROR                VALUE 'C52'.
          88  MQGET-ERROR                 VALUE 'C53'.
          88  MQPUT-ERROR                 VALUE 'C54'.
          88  MQPT1-ERROR                 VALUE 'C55'.
          88  MQCLOSE-ERROR               VALUE 'C56'.
          88  MQDISC-ERROR                VALUE 'C57'.
*---------------------- CONFIG FILE ERRORS
          88  CNFG-RETURN-READ            VALUE 'C60'.
          88  CNFG-RETURN-WRITE           VALUE 'C61'.
          88  CNFG-CHECKPOINT-ERROR       VALUE 'C65'.
*---------------------- NEGOTIATE ERRORS
          88  NEGOTIATE-RESP-ERROR        VALUE 'C70'.
          88  REQUEST-NOT-ACCEPT-ERROR    VALUE 'C71'.
          88  INIT-DATA-NOT-REC-ERROR     VALUE 'C72'.
*------------------INFORMATIONAL MESSAGES
          88  INFO-CHANNEL-CONNECT        VALUE 'C75'.
          88  INFO-CHANNEL-OPENED         VALUE 'C76'.
          88  INFO-QUEUE-OPENED           VALUE 'C77'.
          88  INFO-LU62-CONNECT           VALUE 'C78'.
          88  INFO-RECEIVER-ALLOC         VALUE 'C79'.
          88  INFO-QUEUE-EMPTY            VALUE 'C80'.
          88  INFO-QUEUE-CLOSED           VALUE 'C81'.
          88  INFO-CHANNEL-DISC           VALUE 'C82'.
          88  INFO-CHANNEL-SHUT           VALUE 'C83'.
          88  INFO-CHANNEL-SHUT-SENT      VALUE 'C83'.
*---------------------- GENERAL CICS ERRORS
          88  RECV-RETURN-LOGIN-EIB        VALUE 'C96'.
          88  EIBRC-RETURN-CODE            VALUE 'C97'.
          88  FATAL-RETURN-CODE            VALUE 'C99'.
*-- --REDEFINES ERROR CODE
        10 ERROR-CODE-REDEF   REDEFINES ERROR-CODE.
           12  ERROR-CODE-TYPE   PIC X.
           12  ERROR-CODE-NUM    PIC 99.
*--REST OF ERROR LINE  ----------------
        10 FILLER               PIC X(5) VALUE ' PRG:'.
        10 ERROR-PROGRAM        PIC X(8) VALUE SPACES.
        10 FILLER               PIC X(5) VALUE ' TRN:'.
        10 ERROR-TRANID         PIC X(4) VALUE SPACES.
        10 FILLER               PIC X(5) VALUE ' TSK:'.
        10 ERROR-TASKNO         PIC 9(5) VALUE ZEROS.
        10 FILLER               PIC X(1) VALUE SPACE.
        10 ERR-FORMATTED-DATE.
           12 ERR-FORMATTED-MM   PIC XX   VALUE SPACES.
           12 FILLER             PIC X    VALUE '/'.
           12 ERR-FORMATTED-DD   PIC XX   VALUE SPACES.
           12 FILLER             PIC X    VALUE '/'.
           12 ERR-FORMATTED-YY   PIC XX   VALUE SPACES.
        10 FILLER               PIC X(1) VALUE SPACE.
        10 ERR-FORMATTED-TIME.
           12 ERR-FORMATTED-HH   PIC XX   VALUE SPACES.
           12 FILLER             PIC X    VALUE ':'.
           12 ERR-FORMATTED-MIN  PIC XX   VALUE SPACES.
           12 FILLER             PIC X    VALUE ':'.
           12 ERR-FORMATTED-SS   PIC XX   VALUE SPACES.
        10 FILLER               PIC X(23) VALUE SPACES.
     05 ERR-MSG-2               PIC X(79) VALUE SPACES.
     05 ERR-MSG-3               PIC X(79) VALUE SPACES.
```

```
     05 ERR-MSG-4.
        10 FILLER                PIC X(9) VALUE 'EIBFN:'.
        10 ERR-DEBUG-EIBFN       PIC X(4) VALUE SPACES.
        10 FILLER                PIC X(4) VALUE SPACES.
        10 FILLER                PIC X(14) VALUE '   EIBRCODE: '.
        10 ERR-DEBUG-EIBRCODE-1  PIC X(4) VALUE SPACES.
        10 ERR-DEBUG-EIBRCODE-2  PIC X(8) VALUE SPACES.
        10 FILLER                PIC X(12) VALUE ' EXEC LINE: '.
        10 ERR-DEBUG-EXEC-LN     PIC X(5) VALUE SPACES.
        10 FILLER                PIC X(19) VALUE SPACES.
     05 ERR-MSG-5.
        10 FILLER                PIC X(9) VALUE 'EIBRESP: '.
        10 ERR-DEBUG-EIBRESP     PIC 9(8) VALUE ZEROS.
        10 FILLER                PIC X(14) VALUE '   EIBRESP2: '.
        10 ERR-DEBUG-EIBRESP2    PIC 9(8) VALUE ZEROS.
        10 FILLER                PIC X(4) VALUE SPACES.
        10 FILLER                PIC X(12) VALUE ' EIBRSRCE: '.
        10 ERR-DEBUG-EIBRSRCE    PIC X(8) VALUE SPACES.
        10 FILLER                PIC X(10) VALUE '  ABCODE: '.
        10 ERR-DEBUG-ABEND       PIC X(4) VALUE SPACES.
        10 FILLER                PIC X(2) VALUE SPACES.
 01  FILLER REDEFINES ERR-MESSAGE.
     05 ERR-MESSAGE-LN           PIC X(79) OCCURS 5.
*--COMMON ERR-MSG-2 MESSAGES
 01  ERR-MSG-QID.
     05 FILLER                   PIC X(15) VALUE
        'QUEUE ID  :'.
     05 MSG-QID-KEY              PIC X(48) VALUE SPACES.
 01  ERR-MSG-CID.
     05 FILLER                   PIC X(15) VALUE
        'CHANNEL ID:'.
     05 MSG-QID-KEY              PIC X(48) VALUE SPACES.
*-------------------------------------------------------------*
  EJECT
 77  WS-DATA-LENGTH                PIC S9(4) COMP VALUE ZERO.
 01  WS-DATA-ALL.
     05  WS-DATA-WITH-QUEUE.
       10  WS-DATA-WITH-TIMES.
          12  WS-DATA-WITH-FUNCTION.
            15  WS-DATA-FUNCTION    PIC  X(4)  VALUE 'PUT'.
             88 WS-PUT                         VALUE 'PUT'.
             88 WS-GET                         VALUE 'GET'.
          12  FILLER                PIC  X     VALUE ' '.
          12  WS-DATA-TIMES         PIC  99    VALUE 01.
       10  WS-DATA-SYNC-FLAG        PIC  X     VALUE ' '.
       10  WS-DATA-QUEUE            PIC  X(48) VALUE SPACES.
     EJECT
*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
 01  WS-ALL-MSG.
     05  WS-OK-MSG.
         10  FILLER                   PIC  X(80) VALUE
          ' FULL CYCLE HAS BEEN PERFORMED SUCCESSFULLY'.
         10 WS-OK-MSG-1               PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-2               PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-3               PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-4               PIC  X(80) VALUE SPACES.
         10 WS-OK-MSG-5               PIC  X(80) VALUE SPACES.
```

```
         10 WS-OK-MSG-6              PIC  X(80) VALUE SPACES.
     05  WS-ERR-LINES.
         10  FILLER                  PIC  X(400) VALUE SPACES.
 01  WS-OK-STATS-LINE-1.
     05  FILLER                  PIC  X(20) VALUE
        '    QUEUE USED -'.
     05  WS-OK-QUEUE             PIC  X(48).
 01  WS-OK-STATS-LINE-2.
     05  FILLER                  PIC  X(20) VALUE
        ' REPLY Q-'.
     05  WS-OK-QUEUE-REPLY       PIC  X(48).
 01  WS-OK-STATS-LINE-3.
     05  FILLER                  PIC  X(40) VALUE
        '    NUMBER OF MESSAGES PROCESSED -'.
     05  WS-OK-MESSAGES          PIC Z99.
 01  WS-OK-STATS-LINE-4.
     05  FILLER                  PIC  X(40) VALUE
        '    TOTAL SECONDS ..... -'.
     05  WS-OK-TIME              PIC  X(8).
*-----------------------------------------------------------------*
  EJECT
*-----------------------------------------------------------------*
 01  WS-ERROR-MESSAGES.
     05 WS-ERR-DATA.
       10  FILLER                   PIC  X(13) VALUE
       '  DATA ERROR:'.
       10  FILLER                   PIC  X(9) VALUE
       '  LENGTH='.
       10  WS-ERR-DATA-LENGTH       PIC  9(8) VALUE ZERO.
       10  FILLER                   PIC  X(9) VALUE
       ', DATA  ='.
       10  WS-ERR-DATA-AREA         PIC  X(200) VALUE SPACES.
       10  FILLER                   PIC  X(4) VALUE
       '****'.
     05 WS-ERR-DISPLAY.
       10  FILLER                   PIC  X(13) VALUE
       '  MQ   ERROR:'.
       10  FILLER                   PIC  X(9) VALUE
       '  LEVEL ='.
       10  WS-LEVEL                 PIC  X(8) VALUE SPACES.
       10  FILLER                   PIC  X(9) VALUE
       ', FUNC  ='.
       10  WS-FUNCTION              PIC  X(8) VALUE SPACES.
       10  FILLER                   PIC  X(9) VALUE
       ', CC    ='.
       10  WS-ERR-DISPLAY-CCODE     PIC  9(4) VALUE ZERO.
       10  FILLER                   PIC  X(9) VALUE
       ', RC    ='.
       10  WS-ERR-DISPLAY-RCODE     PIC  9(4) VALUE ZERO.
       10  FILLER                   PIC  X(4) VALUE
       '****'.
  EJECT
*-----------------------------------------------------------------*
 01  MQI-VALUES.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQV                                     **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQI constants        **
```

```
**                                                               **
**  %PRODUCT NUMBER                                              **
**                                                               **
**  %COPYRIGHT                                                   **
**                                                               **
**  FUNCTION:         This file declares the constants           **
**                    which form part of the IBM Message         **
**                    Queue Interface (MQI).                     **
**                                                               **
*****************************************************************
*****************************************************************
**  Values Related to MQGMO Structure                            **
*****************************************************************
**   Structure Identifier
  10 MQGMO-STRUC-ID PIC X(4) VALUE 'GMO '.
**   Structure Version Number
  10 MQGMO-VERSION-1 PIC S9(9) COMP  VALUE 1.
**   Get-Message Options
  10 MQGMO-WAIT                 PIC S9(9) COMP  VALUE 1.
  10 MQGMO-NO-WAIT              PIC S9(9) COMP  VALUE 0.
  10 MQGMO-BROWSE-FIRST         PIC S9(9) COMP  VALUE 16.
  10 MQGMO-BROWSE-NEXT          PIC S9(9) COMP  VALUE 32.
  10 MQGMO-ACCEPT-TRUNCATED-MSG PIC S9(9) COMP  VALUE 64.
  10 MQGMO-SET-SIGNAL           PIC S9(9) COMP  VALUE 8.
  10 MQGMO-SYNCPOINT            PIC S9(9) COMP  VALUE 2.
  10 MQGMO-NO-SYNCPOINT         PIC S9(9) COMP  VALUE 4.
  10 MQGMO-MSG-UNDER-CURSOR     PIC S9(9) COMP  VALUE 256.
**   Wait Interval
  10 MQWI-UNLIMITED PIC S9(9) COMP  VALUE -1.
*****************************************************************
**  Values Related to MQMD Structure                             **
*****************************************************************
**   Structure Identifier
  10 MQMD-STRUC-ID PIC X(4) VALUE 'MD  '.
**   Structure Version Number
  10 MQMD-VERSION-1 PIC S9(9) COMP  VALUE 1.
**   Report Options
  10 MQRO-NONE PIC S9(9) COMP  VALUE 0.
**   Message Types
  10 MQMT-REQUEST  PIC S9(9) COMP  VALUE 1.
  10 MQMT-REPLY    PIC S9(9) COMP  VALUE 2.
  10 MQMT-DATAGRAM PIC S9(9) COMP  VALUE 8.
  10 MQMT-REPORT   PIC S9(9) COMP  VALUE 4.
**   Expiry Value
  10 MQEI-UNLIMITED PIC S9(9) COMP  VALUE -1.
**   Feedback Values
  10 MQFB-NONE         PIC S9(9) COMP  VALUE 0.
  10 MQFB-QUIT         PIC S9(9) COMP  VALUE 256.
  10 MQFB-SYSTEM-FIRST PIC S9(9) COMP  VALUE 1.
  10 MQFB-SYSTEM-LAST  PIC S9(9) COMP  VALUE 65535.
  10 MQFB-APPL-FIRST   PIC S9(9) COMP  VALUE 65536.
  10 MQFB-APPL-LAST    PIC S9(9) COMP  VALUE 999999999.
**   Encoding Value
  10 MQENC-NATIVE PIC S9(9) COMP  VALUE 785.
**   Encoding Masks
  10 MQENC-INTEGER-MASK  PIC S9(9) COMP  VALUE 15.
  10 MQENC-DECIMAL-MASK  PIC S9(9) COMP  VALUE 240.
  10 MQENC-FLOAT-MASK    PIC S9(9) COMP  VALUE 3840.
  10 MQENC-RESERVED-MASK PIC S9(9) COMP  VALUE -4096.
```

```
**   Encodings for Binary Integers
  10 MQENC-INTEGER-UNDEFINED PIC S9(9) COMP  VALUE 0.
  10 MQENC-INTEGER-NORMAL    PIC S9(9) COMP  VALUE 1.
  10 MQENC-INTEGER-REVERSED  PIC S9(9) COMP  VALUE 2.
**   Encodings for Packed-Decimal Integers
  10 MQENC-DECIMAL-UNDEFINED PIC S9(9) COMP  VALUE 0.
  10 MQENC-DECIMAL-NORMAL    PIC S9(9) COMP  VALUE 16.
  10 MQENC-DECIMAL-REVERSED  PIC S9(9) COMP  VALUE 32.
**   Encodings for Floating-Point Numbers
  10 MQENC-FLOAT-UNDEFINED     PIC S9(9) COMP  VALUE 0.
  10 MQENC-FLOAT-IEEE-NORMAL   PIC S9(9) COMP  VALUE 256.
  10 MQENC-FLOAT-IEEE-REVERSED PIC S9(9) COMP  VALUE 512.
  10 MQENC-FLOAT-S390          PIC S9(9) COMP  VALUE 768.
**   Coded Character-Set Identifier
  10 MQCCSI-Q-MGR PIC S9(9) COMP  VALUE 0.
**   Persistence Values
  10 MQPER-PERSISTENT           PIC S9(9) COMP  VALUE 1.
  10 MQPER-PERSISTENCE-AS-Q-DEF PIC S9(9) COMP  VALUE 2.
**   Message Id Value
  10 MQMI-NONE PIC X(24) VALUE LOW-VALUES.
**   Correllation Id Value
  10 MQCI-NONE PIC X(24) VALUE LOW-VALUES.
******************************************************************
**  Values Related to MQOD Structure                            **
******************************************************************
**   Structure Identifier
  10 MQOD-STRUC-ID PIC X(4) VALUE 'OD  '.
**   Structure Version Number
  10 MQOD-VERSION-1 PIC S9(9) COMP  VALUE 1.
**   Object Types
  10 MQOT-Q PIC S9(9) COMP  VALUE 1.
******************************************************************
**  Values Related to MQPMO Structure                           **
******************************************************************
**   Structure Identifier
  10 MQPMO-STRUC-ID PIC X(4) VALUE 'PMO '.
**   Structure Version Number
  10 MQPMO-VERSION-1 PIC S9(9) COMP  VALUE 1.
**   Put-Message Options
  10 MQPMO-SYNCPOINT            PIC S9(9) COMP  VALUE 2.
  10 MQPMO-NO-SYNCPOINT         PIC S9(9) COMP  VALUE 4.
******************************************************************
**  Values Related to MQTM Structure                            **
******************************************************************
**   Structure Identifier
  10 MQTM-STRUC-ID PIC X(4) VALUE 'TM  '.
**   Structure Version Number
  10 MQTM-VERSION-1 PIC S9(9) COMP  VALUE 1.
******************************************************************
**  Values Related to MQCLOSE Call                              **
******************************************************************
**   Close Options
  10 MQCO-NONE PIC S9(9) COMP  VALUE 0.
******************************************************************
**  Values Related to MQINQ Call                                **
******************************************************************
**   Character-Attribute Selectors
  10 MQCA-BASE-Q-NAME       PIC S9(9) COMP  VALUE 2002.
  10 MQCA-CREATION-DATE     PIC S9(9) COMP  VALUE 2004.
```

```
  10 MQCA-CREATION-TIME     PIC S9(9) COMP  VALUE 2005.
  10 MQCA-FIRST             PIC S9(9) COMP  VALUE 2001.
  10 MQCA-INITIATION-Q-NAME PIC S9(9) COMP  VALUE 2008.
  10 MQCA-LAST              PIC S9(9) COMP  VALUE 4000.
  10 MQCA-PROCESS-NAME      PIC S9(9) COMP  VALUE 2012.
  10 MQCA-Q-DESC            PIC S9(9) COMP  VALUE 2013.
  10 MQCA-Q-NAME            PIC S9(9) COMP  VALUE 2016.
  10 MQCA-REMOTE-Q-MGR-NAME PIC S9(9) COMP  VALUE 2017.
  10 MQCA-REMOTE-Q-NAME     PIC S9(9) COMP  VALUE 2018.
**   Integer-Attribute Selectors
  10 MQIA-CURRENT-Q-DEPTH   PIC S9(9) COMP  VALUE 3.
  10 MQIA-DEF-PERSISTENCE   PIC S9(9) COMP  VALUE 5.
  10 MQIA-DEFINITION-TYPE   PIC S9(9) COMP  VALUE 7.
  10 MQIA-FIRST             PIC S9(9) COMP  VALUE 1.
  10 MQIA-INHIBIT-GET       PIC S9(9) COMP  VALUE 9.
  10 MQIA-INHIBIT-PUT       PIC S9(9) COMP  VALUE 10.
  10 MQIA-LAST              PIC S9(9) COMP  VALUE 2000.
  10 MQIA-MAX-MSG-LENGTH    PIC S9(9) COMP  VALUE 13.
  10 MQIA-MAX-Q-DEPTH       PIC S9(9) COMP  VALUE 15.
  10 MQIA-OPEN-INPUT-COUNT  PIC S9(9) COMP  VALUE 17.
  10 MQIA-OPEN-OUTPUT-COUNT PIC S9(9) COMP  VALUE 18.
  10 MQIA-Q-TYPE            PIC S9(9) COMP  VALUE 20.
  10 MQIA-SHAREABILITY      PIC S9(9) COMP  VALUE 23.
  10 MQIA-TRIGGER-CONTROL   PIC S9(9) COMP  VALUE 24.
  10 MQIA-TRIGGER-TYPE      PIC S9(9) COMP  VALUE 28.
  10 MQIA-USAGE             PIC S9(9) COMP  VALUE 12.
**   Integer Attribute Value Denoting 'Not Applicable'
  10 MQIAV-NOT-APPLICABLE PIC S9(9) COMP  VALUE -1.
******************************************************************
**  Values Related to MQOPEN Call                               **
******************************************************************
**   Open Options
  10 MQOO-INPUT-SHARED    PIC S9(9) COMP  VALUE 2.
  10 MQOO-INPUT-EXCLUSIVE PIC S9(9) COMP  VALUE 4.
  10 MQOO-BROWSE          PIC S9(9) COMP  VALUE 8.
  10 MQOO-OUTPUT          PIC S9(9) COMP  VALUE 16.
  10 MQOO-INQUIRE         PIC S9(9) COMP  VALUE 32.
******************************************************************
**  Values Related to All Calls                                 **
******************************************************************
**   String Lengths
  10 MQ-CREATION-DATE-LENGTH     PIC S9(9) COMP  VALUE 12.
  10 MQ-CREATION-TIME-LENGTH     PIC S9(9) COMP  VALUE 8.
  10 MQ-PROCESS-APPL-ID-LENGTH   PIC S9(9) COMP  VALUE 256.
  10 MQ-PROCESS-DESC-LENGTH      PIC S9(9) COMP  VALUE 64.
  10 MQ-PROCESS-ENV-DATA-LENGTH  PIC S9(9) COMP  VALUE 128.
  10 MQ-PROCESS-NAME-LENGTH      PIC S9(9) COMP  VALUE 48.
  10 MQ-PROCESS-USER-DATA-LENGTH PIC S9(9) COMP  VALUE 128.
  10 MQ-Q-DESC-LENGTH            PIC S9(9) COMP  VALUE 64.
  10 MQ-Q-NAME-LENGTH            PIC S9(9) COMP  VALUE 48.
  10 MQ-Q-MGR-DESC-LENGTH        PIC S9(9) COMP  VALUE 64.
  10 MQ-Q-MGR-NAME-LENGTH        PIC S9(9) COMP  VALUE 48.
  10 MQ-TRIGGER-DATA-LENGTH      PIC S9(9) COMP  VALUE 64.
**   Completion Codes
  10 MQCC-OK      PIC S9(9) COMP  VALUE 0.
  10 MQCC-WARNING PIC S9(9) COMP  VALUE 1.
  10 MQCC-FAILED  PIC S9(9) COMP  VALUE 2.
**   Reason Codes
  10 MQRC-NONE                      PIC S9(9) COMP  VALUE 0.
```

```
10 MQRC-ACCESS-RESTRICTED         PIC S9(9) COMP  VALUE 2000.
10 MQRC-ALIAS-BASE-Q-TYPE-ERROR   PIC S9(9) COMP  VALUE 2001.
10 MQRC-ALREADY-CONNECTED         PIC S9(9) COMP  VALUE 2002.
10 MQRC-BUFFER-ERROR              PIC S9(9) COMP  VALUE 2004.
10 MQRC-BUFFER-LENGTH-ERROR       PIC S9(9) COMP  VALUE 2005.
10 MQRC-CHAR-ATTR-LENGTH-ERROR    PIC S9(9) COMP  VALUE 2006.
10 MQRC-CHAR-ATTRS-ERROR          PIC S9(9) COMP  VALUE 2007.
10 MQRC-CHAR-ATTRS-TOO-SHORT      PIC S9(9) COMP  VALUE 2008.
10 MQRC-CONNECTION-BROKEN         PIC S9(9) COMP  VALUE 2009.
10 MQRC-DATA-LENGTH-ERROR         PIC S9(9) COMP  VALUE 2010.
10 MQRC-EXPIRY-ERROR              PIC S9(9) COMP  VALUE 2013.
10 MQRC-FEEDBACK-ERROR            PIC S9(9) COMP  VALUE 2014.
10 MQRC-GET-INHIBITED             PIC S9(9) COMP  VALUE 2016.
10 MQRC-HANDLE-NOT-AVAILABLE      PIC S9(9) COMP  VALUE 2017.
10 MQRC-HCONN-ERROR               PIC S9(9) COMP  VALUE 2018.
10 MQRC-HOBJ-ERROR                PIC S9(9) COMP  VALUE 2019.
10 MQRC-INT-ATTR-COUNT-ERROR      PIC S9(9) COMP  VALUE 2021.
10 MQRC-INT-ATTR-COUNT-TOO-SMALL  PIC S9(9) COMP  VALUE 2022.
10 MQRC-INT-ATTRS-ARRAY-ERROR     PIC S9(9) COMP  VALUE 2023.
10 MQRC-MAX-CONNS-LIMIT-REACHED   PIC S9(9) COMP  VALUE 2025.
10 MQRC-MD-ERROR                  PIC S9(9) COMP  VALUE 2026.
10 MQRC-MISSING-REPLY-TO-Q        PIC S9(9) COMP  VALUE 2027.
10 MQRC-MSG-TYPE-ERROR            PIC S9(9) COMP  VALUE 2029.
10 MQRC-MSG-TOO-BIG-FOR-Q         PIC S9(9) COMP  VALUE 2030.
10 MQRC-NO-MSG-AVAILABLE          PIC S9(9) COMP  VALUE 2033.
10 MQRC-NO-MSG-UNDER-CURSOR       PIC S9(9) COMP  VALUE 2034.
10 MQRC-NOT-AUTHORIZED            PIC S9(9) COMP  VALUE 2035.
10 MQRC-NOT-OPEN-FOR-BROWSE       PIC S9(9) COMP  VALUE 2036.
10 MQRC-NOT-OPEN-FOR-INPUT        PIC S9(9) COMP  VALUE 2037.
10 MQRC-NOT-OPEN-FOR-INQUIRE      PIC S9(9) COMP  VALUE 2038.
10 MQRC-NOT-OPEN-FOR-OUTPUT       PIC S9(9) COMP  VALUE 2039.
10 MQRC-OBJECT-CHANGED            PIC S9(9) COMP  VALUE 2041.
10 MQRC-OBJECT-IN-USE             PIC S9(9) COMP  VALUE 2042.
10 MQRC-OBJECT-TYPE-ERROR         PIC S9(9) COMP  VALUE 2043.
10 MQRC-OD-ERROR                  PIC S9(9) COMP  VALUE 2044.
10 MQRC-OPTION-NOT-VALID-FOR-TYPE PIC S9(9) COMP  VALUE 2045.
10 MQRC-OPTIONS-ERROR             PIC S9(9) COMP  VALUE 2046.
10 MQRC-PERSISTENCE-ERROR         PIC S9(9) COMP  VALUE 2047.
10 MQRC-PRIORITY-EXCEEDS-MAXIMUM  PIC S9(9) COMP  VALUE 2049.
10 MQRC-PRIORITY-ERROR            PIC S9(9) COMP  VALUE 2050.
10 MQRC-PUT-INHIBITED             PIC S9(9) COMP  VALUE 2051.
10 MQRC-Q-FULL                    PIC S9(9) COMP  VALUE 2053.
10 MQRC-Q-SPACE-NOT-AVAILABLE     PIC S9(9) COMP  VALUE 2056.
10 MQRC-Q-MGR-NAME-ERROR          PIC S9(9) COMP  VALUE 2058.
10 MQRC-Q-MGR-NOT-AVAILABLE       PIC S9(9) COMP  VALUE 2059.
10 MQRC-REPORT-OPTIONS-ERROR      PIC S9(9) COMP  VALUE 2061.
10 MQRC-SECURITY-ERROR            PIC S9(9) COMP  VALUE 2063.
10 MQRC-SELECTOR-COUNT-ERROR      PIC S9(9) COMP  VALUE 2065.
10 MQRC-SELECTOR-LIMIT-EXCEEDED   PIC S9(9) COMP  VALUE 2066.
10 MQRC-SELECTOR-ERROR            PIC S9(9) COMP  VALUE 2067.
10 MQRC-SELECTOR-NOT-FOR-TYPE     PIC S9(9) COMP  VALUE 2068.
10 MQRC-SIGNAL-OUTSTANDING        PIC S9(9) COMP  VALUE 2069.
10 MQRC-SIGNAL-REQUEST-ACCEPTED   PIC S9(9) COMP  VALUE 2070.
10 MQRC-STORAGE-NOT-AVAILABLE     PIC S9(9) COMP  VALUE 2071.
10 MQRC-SYNCPOINT-NOT-AVAILABLE   PIC S9(9) COMP  VALUE 2072.
10 MQRC-TRUNCATED-MSG-ACCEPTED    PIC S9(9) COMP  VALUE 2079.
10 MQRC-TRUNCATED-MSG-FAILED      PIC S9(9) COMP  VALUE 2080.
10 MQRC-UNEXPECTED-CONNECT-ERROR  PIC S9(9) COMP  VALUE 2081.
10 MQRC-UNKNOWN-ALIAS-BASE-Q      PIC S9(9) COMP  VALUE 2082.
```

```
  10 MQRC-UNKNOWN-OBJECT-NAME       PIC S9(9) COMP  VALUE 2085.
  10 MQRC-UNKNOWN-OBJECT-Q-MGR      PIC S9(9) COMP  VALUE 2086.
  10 MQRC-UNKNOWN-REMOTE-Q-MGR      PIC S9(9) COMP  VALUE 2087.
  10 MQRC-WAIT-INTERVAL-ERROR       PIC S9(9) COMP  VALUE 2090.
  10 MQRC-XMIT-Q-TYPE-ERROR         PIC S9(9) COMP  VALUE 2091.
  10 MQRC-XMIT-Q-USAGE-ERROR        PIC S9(9) COMP  VALUE 2092.
  10 MQRC-PMO-ERROR                 PIC S9(9) COMP  VALUE 2173.
  10 MQRC-GMO-ERROR                 PIC S9(9) COMP  VALUE 2186.
  10 MQRC-UNEXPECTED-ERROR          PIC S9(9) COMP  VALUE 2195.
  10 MQRC-FILE-SYSTEM-ERROR         PIC S9(9) COMP  VALUE 2208.
**********************************************************************
**  Values Related to Queue Attributes                              **
**********************************************************************
**   Queue Types
  10 MQQT-LOCAL  PIC S9(9) COMP  VALUE 1.
  10 MQQT-ALIAS  PIC S9(9) COMP  VALUE 3.
  10 MQQT-REMOTE PIC S9(9) COMP  VALUE 6.
**   Queue Definition Types
  10 MQQDT-PREDEFINED PIC S9(9) COMP  VALUE 1.
**   Inhibit Get
  10 MQQA-GET-INHIBITED PIC S9(9) COMP  VALUE 1.
  10 MQQA-GET-ALLOWED   PIC S9(9) COMP  VALUE 0.
**   Inhibit Put
  10 MQQA-PUT-INHIBITED PIC S9(9) COMP  VALUE 1.
  10 MQQA-PUT-ALLOWED   PIC S9(9) COMP  VALUE 0.
**   Queue Shareability
  10 MQQA-SHAREABLE     PIC S9(9) COMP  VALUE 1.
  10 MQQA-NOT-SHAREABLE PIC S9(9) COMP  VALUE 0.
**   Message Delivery Sequence
  10 MQMDS-FIFO PIC S9(9) COMP  VALUE 1.
**   Trigger Control
  10 MQTC-OFF PIC S9(9) COMP  VALUE 0.
  10 MQTC-ON  PIC S9(9) COMP  VALUE 1.
**   Trigger Types
  10 MQTT-NONE  PIC S9(9) COMP  VALUE 0.
  10 MQTT-FIRST PIC S9(9) COMP  VALUE 1.
  10 MQTT-EVERY PIC S9(9) COMP  VALUE 2.
**   Queue Usage
  10 MQUS-NORMAL       PIC S9(9) COMP  VALUE 0.
  10 MQUS-TRANSMISSION PIC S9(9) COMP  VALUE 1.
**********************************************************************
**  Values Related to Process-Definition Attributes                 **
**********************************************************************
**   Application Type
  10 MQAT-USER-FIRST PIC S9(9) COMP  VALUE 65536.
  10 MQAT-USER-LAST  PIC S9(9) COMP  VALUE 999999999.
  10 MQAT-OS2        PIC S9(9) COMP VALUE 4.
  10 MQAT-DOS        PIC S9(9) COMP VALUE 5.
  10 MQAT-AIX        PIC S9(9) COMP VALUE 6.
  10 MQAT-OS400      PIC S9(9) COMP VALUE 8.
  10 MQAT-WINDOWS    PIC S9(9) COMP VALUE 9.
  10 MQAT-CICS-VSE   PIC S9(9) COMP VALUE 10.
  10 MQAT-VMS        PIC S9(9) COMP VALUE 12.
  10 MQAT-GUARDIAN   PIC S9(9) COMP VALUE 13.
  10 MQAT-VOS        PIC S9(9) COMP VALUE 14.
**********************************************************************
**  Values Related to Queue-Manager Attributes                      **
**********************************************************************
**   Syncpoint Availability
```

```
  10 MQSP-AVAILABLE PIC S9(9) COMP  VALUE 1.
  EJECT
*-----------------------------------------------------------------*
* ENVIRONMENT VALUES
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
* - BEGIN -      ***  COPYBOOK: MQIENV    ***          - BEGIN - *
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
*    MQSERIES EZBRIDGE TRANSACT ENVIRONMENT                       *
*-----------------------------------------------------------------*
 01  ENVIRONMENT-INFO.
     03 ENV-DEFINITION.
        05 ENV-CONFIG-DDNAME           PIC  X(8) VALUE 'MQFCNFG'.
        05 ENV-SYSTEM-NUMBER           PIC S9(4) COMP VALUE +1.
        05 FILLER                      PIC  XX   VALUE SPACES.
        05 ENV-MASTER-TERMINAL-INFO.
           10 ENV-MT-MASTER-TASK-ID    PIC  X(4) VALUE 'MQMT'.
           10 ENV-MT-MASTER-PROGRAM    PIC  X(8) VALUE 'MQPMTP'.
           10 ENV-MT-MASTER-MAPSCREEN  PIC  X(8) VALUE 'MQMMTP'.
           10 ENV-MT-CONFIG-TASK-ID    PIC  X(4) VALUE 'MQMC'.
           10 ENV-MT-CONFIG-PROGRAM    PIC  X(8) VALUE 'MQPMCFG'.
           10 ENV-MT-CONFIG-MAPSCREEN  PIC  X(8) VALUE 'MQMMCFG'.
           10 ENV-MT-MONITOR-TASK-ID   PIC  X(4) VALUE 'MQMM'.
           10 ENV-MT-MONITOR-PROGRAM   PIC  X(8) VALUE 'MQPMMON'.
           10 ENV-MT-MONITOR-MAPSCREEN PIC  X(8) VALUE 'MQMMMON'.
           10 ENV-MT-OPER-TASK-ID      PIC  X(4) VALUE 'MQMO'.
           10 ENV-MT-OPER-PROGRAM      PIC  X(8) VALUE 'MQPMOPR'.
           10 ENV-MT-OPER-MAPSCREEN    PIC  X(8) VALUE 'MQMMOPR'.
           10 ENV-MT-DISP-TASK-ID      PIC  X(4) VALUE 'MQBQ'.
           10 ENV-MT-DISP-PROGRAM      PIC  X(8) VALUE 'MQPDISP'.
           10 ENV-MT-DISP-MAPSCREEN    PIC  X(8) VALUE 'MQMDISP'.
           10 ENV-MT-QUEUE-TASK-ID     PIC  X(4) VALUE 'MQMQ'.
           10 ENV-MT-QUEUE-PROGRAM     PIC  X(8) VALUE 'MQPMQUE'.
           10 ENV-MT-QUEUE-MAPSCREEN   PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-QUEUEI-TASK-ID    PIC  X(4) VALUE 'MQDQ'.
           10 ENV-MT-QUEUEI-PROGRAM    PIC  X(8) VALUE 'MQPMQUE'.
           10 ENV-MT-QUEUEI-MAPSCREEN  PIC  X(8) VALUE 'MQMMQUE'.
           10 ENV-MT-CHAN-TASK-ID      PIC  X(4) VALUE 'MQMH'.
           10 ENV-MT-CHAN-PROGRAM      PIC  X(8) VALUE 'MQPMCHN'.
           10 ENV-MT-CHAN-MAPSCREEN    PIC  X(8) VALUE 'MQMMCHN'.
           10 ENV-MT-CHANI-TASK-ID     PIC  X(4) VALUE 'MQDH'.
           10 ENV-MT-CHANI-PROGRAM     PIC  X(8) VALUE 'MQPMCHN'.
           10 ENV-MT-CHANI-MAPSCREEN   PIC  X(8) VALUE 'MQMMCHN'.
           10 ENV-MT-SYS-TASK-ID       PIC  X(4) VALUE 'MQMS'.
           10 ENV-MT-SYS-PROGRAM       PIC  X(8) VALUE 'MQPMSYS'.
           10 ENV-MT-SYS-MAPSCREEN     PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-SYSI-TASK-ID      PIC  X(4) VALUE 'MQDS'.
           10 ENV-MT-SYSI-PROGRAM      PIC  X(8) VALUE 'MQPMSYS'.
           10 ENV-MT-SYSI-MAPSCREEN    PIC  X(8) VALUE 'MQMMSYS'.
           10 ENV-MT-MONQ-TASK-ID      PIC  X(4) VALUE 'MQQM'.
           10 ENV-MT-MONQ-PROGRAM      PIC  X(8) VALUE 'MQPMMOQ'.
           10 ENV-MT-MONQ-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOQ'.
           10 ENV-MT-MONC-TASK-ID      PIC  X(4) VALUE 'MQCM'.
           10 ENV-MT-MONC-PROGRAM      PIC  X(8) VALUE 'MQPMMOC'.
           10 ENV-MT-MONC-MAPSCREEN    PIC  X(8) VALUE 'MQMMMOC'.
           10 ENV-MT-SS-TASK-ID        PIC  X(4) VALUE 'MQMA'.
           10 ENV-MT-SS-PROGRAM        PIC  X(8) VALUE 'MQPMSS'.
           10 ENV-MT-SS-MAPSCREEN      PIC  X(8) VALUE 'MQMMSS'.
```

```
           10 ENV-MT-SC-TASK-ID        PIC  X(4) VALUE 'MQMB'.
           10 ENV-MT-SC-PROGRAM        PIC  X(8) VALUE 'MQPMSC'.
           10 ENV-MT-SC-MAPSCREEN      PIC  X(8) VALUE 'MQMMSC'.
           10 ENV-MT-SI-TASK-ID        PIC  X(4) VALUE 'MQMI'.
           10 ENV-MT-SI-PROGRAM        PIC  X(8) VALUE 'MQPMSI'.
           10 ENV-MT-SI-MAPSCREEN      PIC  X(8) VALUE 'MQMMSI'.
           10 ENV-MT-SR-TASK-ID        PIC  X(4) VALUE 'MQMR'.
           10 ENV-MT-SR-PROGRAM        PIC  X(8) VALUE 'MQPMMSN'.
           10 ENV-MT-SR-MAPSCREEN      PIC  X(8) VALUE 'MQMMMSN'.
           10 ENV-MT-SD-TASK-ID        PIC  X(4) VALUE 'MQMD'.
           10 ENV-MT-SD-PROGRAM        PIC  X(8) VALUE 'MQPMDEL'.
           10 ENV-MT-SD-MAPSCREEN      PIC  X(8) VALUE 'MQMMDEL'.
        05 ENV-INDEPENDENT-ITEMS.
           10 ENV-II-ERROR-TD          PIC  X(4) VALUE 'MQER'.
           10 ENV-II-ERROR-CSMT        PIC  X(4) VALUE 'CSMT'.
           10 ENV-II-MONITOR           PIC  X(4) VALUE 'MQSM'.
           10 ENV-II-START-STOP        PIC  X(4) VALUE 'MQSS'.
           10 ENV-II-SYSTEM-ANCHOR     PIC  X(8) VALUE 'MQTAQM'.
           10 ENV-II-SYSTEM-PREFIX     PIC  X(4) VALUE 'MQI '.
           10 ENV-II-DUMPCODE          PIC  X(4) VALUE 'MQ??'.
           10 ENV-II-LINK-ERROR        PIC  X(8) VALUE 'MQPERR  '.
           10 ENV-II-LINK-EIB1         PIC  X(8) VALUE 'MQPEIB1 '.
           10 ENV-II-LINK-AIP0         PIC  X(8) VALUE 'MQPAIP0 '.
           10 ENV-II-LINK-AIP1         PIC  X(8) VALUE 'MQPAIP1 '.
           10 ENV-II-TRAN-AIP2         PIC  X(4) VALUE 'MQ02'.
           10 ENV-II-LINK-AIP2         PIC  X(8) VALUE 'MQPAIP2 '.
           10 ENV-II-LINK-ECHO         PIC  X(8) VALUE 'MQPECHO '.
           10 ENV-II-LINK-FINDQ        PIC  X(8) VALUE 'MQPFINDQ'.
           10 ENV-II-LINK-QUE1         PIC  X(8) VALUE 'MQPQUE1 '.
           10 ENV-II-LINK-QUE2         PIC  X(8) VALUE 'MQPQUE2 '.
           10 ENV-II-LINK-INIT1        PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-ENQ-INIT1         PIC  X(8) VALUE 'MQPINIT1'.
           10 ENV-II-LINK-INIT2        PIC  X(8) VALUE 'MQPINIT2'.
           10 ENV-II-LINK-SSQ          PIC  X(8) VALUE 'MQPSSQ  '.
           10 ENV-II-LINK-SCHK         PIC  X(8) VALUE 'MQPSCHK '.
           10 ENV-II-LINK-SENDER       PIC  X(8) VALUE 'MQPSEND '.
           10 ENV-II-LINK-RECIEVER     PIC  X(8) VALUE 'MQPRECV '.
           10 ENV-II-TRAN-CHAN-CHECKP  PIC  X(4) VALUE 'MQCP'.
           10 ENV-II-LINK-CHAN-CHECKP  PIC  X(8) VALUE 'MQPCCKPT'.
           10 ENV-II-TRAN-QUE-DELETE   PIC  X(4) VALUE 'MQQD'.
           10 ENV-II-TRAN-QUE-DEL-ALL  PIC  X(4) VALUE 'MQQA'.
           10 ENV-II-LINK-QUE-DELETE   PIC  X(8) VALUE 'MQPQDEL'.
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
     EJECT
*-----------------------------------------------------------------*
* COMMON PARMS
 01  FILLER          PIC X(8) VALUE 'PARMS:--'.
 01  WS-HCONN-ADDR-AREA.
     05  WS-HCONN-VALUE             USAGE POINTER.
 01  WS-HOBJ-ADDR-AREA.
     05  WS-HOBJ-VALUE              USAGE POINTER.
 01  WS-HOBJ-ADDR-AREA-REPLY.
     05  WS-HOBJ-VALUE-REPLY        USAGE POINTER.
 01  WS-CCODE-ADDR-AREA.
     05  WS-CCODE-VALUE             PIC S9(8) COMP.
 01  WS-RCODE-ADDR-AREA.
     05  WS-RCODE-VALUE             PIC S9(8) COMP.
*-----------------------------------------------------------------*
```

```
*--CONNECT PARM
 01  WS-QM-NAME-AREA.
     05  WS-QM-NAME-CONNECT         PIC  X(48).
*--OPEN    PARM
 01  WS-Q-NAME-AREA.
******************************************************************
**                                                              **
**  FILE NAME:        CMQODV                                    **
**                                                              **
**  DESCRIPTIVE NAME: COBOL copy file for MQOD structure        **
**                                                              **
**  %PRODUCT NUMBER                                           **
**                                                              **
**  %COPYRIGHT                                                **
**                                                              **
**  FUNCTION:         This file declares the MQOD structure,    **
**                    which forms part of the IBM Message       **
**                    Queue Interface (MQI).                    **
**                                                              **
******************************************************************
**   MQOD structure
  10 MQOD.
**    Structure identifier
   15 MQOD-STRUCID         PIC X(4) VALUE 'OD  '.
**    Structure version number
   15 MQOD-VERSION         PIC S9(9) COMP   VALUE 1.
**    Object type
   15 MQOD-OBJECTTYPE      PIC S9(9) COMP   VALUE 1.
**    Object name
   15 MQOD-OBJECTNAME      PIC X(48) VALUE SPACES.
**    Object queue manager name
   15 MQOD-OBJECTQMGRNAME  PIC X(48) VALUE SPACES.
**    Dynamic queue name
   15 MQOD-DYNAMICQNAME    PIC X(48) VALUE '*'.
   15 FILLER               PIC X(12) VALUE ' '.
 01  WS-Q-OPEN-OPTIONS.
     05  WS-Q-OPEN-OPTIONS-VALUE    PIC S9(8) COMP.
     EJECT
*--INQ
 01  MQI-SECTOR-COUNT.
     05  WS-SECTOR-COUNT            PIC S9(8) COMP.
 01  MQI-SECTOR.
     05  WS-SECTOR                  PIC XXXX.
 01  MQI-IN-ATTR-COUNT.
     05  WS-IN-ATTR-COUNT           PIC S9(8) COMP.
 01  MQI-IN-ATTR.
     05  WS-IN-ATTR                 PIC XXXX.
 01  MQI-CHAR-ATTR-LENGTH.
     05  WS-CHAR-ATTR-LENGTH        PIC S9(8) COMP.
 01  MQI-CHAR-ATTR.
     05  WS-CHAR-ATTR               PIC XXXX.
*--PUT/GET PARM
 01  WS-MSG-DESCRIPTOR.
******************************************************************
**                                                              **
**  FILE NAME:        CMQMDV                                    **
**                                                              **
**  DESCRIPTIVE NAME: COBOL copy file for MQMD structure        **
**                                                              **
```

```
**  %PRODUCT NUMBER                                                **
**                                                                 **
**  %COPYRIGHT                                                     **
**                                                                 **
**  FUNCTION:         This file declares the MQMD structure,       **
**                    which forms part of the IBM Message          **
**                    Queue Interface (MQI).                       **
**                                                                 **
*********************************************************************
**   MQMD structure
  10 MQMD.
**    Structure identifier
   15 MQMD-STRUCID           PIC X(4) VALUE 'MD  '.
**    Structure version number
   15 MQMD-VERSION           PIC S9(9) COMP  VALUE 1.
**    Reserved
   15 MQMD-REPORT            PIC S9(9) COMP  VALUE 0.
**    Message type
   15 MQMD-MSGTYPE           PIC S9(9) COMP  VALUE 8.
**    Reserved
   15 MQMD-EXPIRY            PIC S9(9) COMP  VALUE -1.
**    Feedback code
   15 MQMD-FEEDBACK          PIC S9(9) COMP  VALUE 0.
**    Data encoding
   15 MQMD-ENCODING          PIC S9(9) COMP  VALUE 785.
**    Coded character set identifier
   15 MQMD-CODEDCHARSETID    PIC S9(9) COMP  VALUE 0.
**    Format name
   15 MQMD-FORMAT            PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PRIORITY          PIC S9(9) COMP  VALUE 0.
**    Message persistence
   15 MQMD-PERSISTENCE       PIC S9(9) COMP  VALUE 2.
**    Message identifier
   15 MQMD-MSGID             PIC X(24) VALUE LOW-VALUES.
**    Correlation identifier
   15 MQMD-CORRELID          PIC X(24) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-BACKOUTCOUNT      PIC S9(9) COMP  VALUE 0.
**    Name of reply queue
   15 MQMD-REPLYTOQ          PIC X(48) VALUE SPACES.
**    Name of reply queue manager
   15 MQMD-REPLYTOQMGR       PIC X(48) VALUE SPACES.
**    Reserved
   15 MQMD-USERIDENTIFIER    PIC X(12) VALUE SPACES.
**    Reserved
   15 MQMD-ACCOUNTINGTOKEN   PIC X(32) VALUE LOW-VALUES.
**    Reserved
   15 MQMD-APPLIDENTITYDATA  PIC X(32) VALUE SPACES.
**    Reserved
   15 MQMD-PUTAPPLTYPE       PIC S9(9) COMP  VALUE 0.
**    Reserved
   15 MQMD-PUTAPPLNAME       PIC X(28) VALUE SPACES.
**    Reserved
   15 MQMD-PUTDATE           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-PUTTIME           PIC X(8) VALUE SPACES.
**    Reserved
   15 MQMD-APPLORIGINDATA    PIC X(4) VALUE SPACES.
```

```
 01  WS-PUT-OPTIONS.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQPMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQPMO structure      **
**                                                             **
**  %PRODUCT NUMBER                                          **
**                                                             **
**  %COPYRIGHT                                               **
**                                                             **
**  FUNCTION:         This file declares the MQPMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************
**   MQPMO structure
  10 MQPMO.
**    Structure identifier
   15 MQPMO-STRUCID           PIC X(4) VALUE 'PMO '.
**    Structure version number
   15 MQPMO-VERSION           PIC S9(9) COMP   VALUE 1.
**    Reserved
   15 MQPMO-OPTIONS           PIC S9(9) COMP   VALUE 0.
**    Reserved
   15 MQPMO-TIMEOUT           PIC S9(9) COMP   VALUE -1.
**    Reserved
   15 MQPMO-CONTEXT           PIC S9(9) COMP   VALUE 0.
**    Reserved
   15 MQPMO-KNOWNDESTCOUNT    PIC S9(9) COMP   VALUE 0.
**    Reserved
   15 MQPMO-UNKNOWNDESTCOUNT  PIC S9(9) COMP   VALUE 0.
**    Reserved
   15 MQPMO-INVALIDDESTCOUNT  PIC S9(9) COMP   VALUE 0.
**    Resolved name of destination queue
   15 MQPMO-RESOLVEDQNAME     PIC X(48) VALUE SPACES.
**    Resolved name of destination queue manager
   15 MQPMO-RESOLVEDQMGRNAME  PIC X(48) VALUE SPACES.
 01  WS-GET-OPTIONS.
*****************************************************************
**                                                             **
**  FILE NAME:        CMQGMOV                                  **
**                                                             **
**  DESCRIPTIVE NAME: COBOL copy file for MQGMO structure      **
**                                                             **
**  %PRODUCT NUMBER                                          **
**                                                             **
**  %COPYRIGHT                                               **
**                                                             **
**  FUNCTION:         This file declares the MQGMO structure,  **
**                    which forms part of the IBM Message      **
**                    Queue Interface (MQI).                   **
**                                                             **
*****************************************************************
**   MQGMO structure
  10 MQGMO.
**    Structure identifier
   15 MQGMO-STRUCID        PIC X(4) VALUE 'GMO '.
**    Structure version number
```

```
   15 MQGMO-VERSION        PIC S9(9) COMP   VALUE 1.
**    Options
   15 MQGMO-OPTIONS        PIC S9(9) COMP   VALUE 0.
**    Wait interval
   15 MQGMO-WAITINTERVAL   PIC S9(9) COMP   VALUE 0.
**    Signal
   15 MQGMO-SIGNAL1        PIC S9(9) COMP   VALUE 0.
**    Reserved
   15 MQGMO-SIGNAL2        PIC S9(9) COMP   VALUE 0.
**    Resolved name of destination queue
   15 MQGMO-RESOLVEDQNAME  PIC X(48) VALUE SPACES.
 01  WS-DATA-L-AREA.
     05  WS-DATA-LENGTH-USER        PIC S9(8) COMP VALUE +200.
 01  WS-BUFFER-L-AREA.
     05  WS-BUFFER-LENGTH           PIC S9(8) COMP VALUE +200.
 77  WS-MSG-LENGTH                  PIC S9(8) COMP VALUE +200.
 01  WS-MSG-AREA.
     05  FILLER                     PIC  X(500) VALUE
     'THIS IS A MESSAGE TEXT'.
 01  WS-BUFFER-AREA.
     05  WS-BUFFER-TS                PIC  X(16)  VALUE SPACES.
     05  WS-BUFFER-TEXT              PIC  X(500) VALUE SPACES.
 01  WS-ORIGINAL-BUFFER-AREA.
     05  FILLER                     PIC  X(200) VALUE
     'THIS IS A MESSAGE TEXT'.
*-----------------------------------------------------------------*
  EJECT
*-----------------------------------------------------------------*
 01  LK-DATA.
     05  FILLER                   PIC X(1000).
  EJECT
*-----------------------------------------------------------------*
 PROCEDURE DIVISION.
*-----------------------------------------------------------------*
 0000-MAIN-LINE.
*--INITIALIZE
     MOVE  'INIT   ' TO   WS-LEVEL.
     PERFORM 1000-INITIALIZE
        THRU 1000-EXIT.
*-----------------------------------------------------------------*
     PERFORM    1   TIMES
*--SEND QUEUE RECORDS
          IF WS-PUT
           THEN
               PERFORM 2000-PUT-MESSAGES
                  THRU 2000-EXIT
          END-IF
*--GET  QUEUE RECORDS
          IF WS-GET
           THEN
               PERFORM 3000-GET-MESSAGES
                  THRU 3000-EXIT
          END-IF
     END-PERFORM.
*-----------------------------------------------------------------*
 0000-SEND-TOTALS.
     IF NOT WS-STARTED
      THEN
             PERFORM   7000-SEND-TOTALS.
```

```
*-------------------------------------------------------------*
 0000-RETURN.
*    EXEC CICS RETURN
*         END-EXEC.

     CALL 'EZBMBTCH'  USING  EZDISC.
     MOVE ZEROES TO RETURN-CODE.
     GOBACK.
  EJECT
*-------------------------------------------------------------*
 1000-INITIALIZE.
*-------------------------------------------------------------*
* PURPOSE: SETUP DATA AREAS
*-------------------------------------------------------------*
     ACCEPT WORK2  FROM TIME.
     ACCEPT WS-DATE-YYMMDD FROM DATE.
     IF WS-DATE-YY >  50
      THEN
            MOVE  19                 TO  WS-DATE-CC
      ELSE
            MOVE  20                 TO  WS-DATE-CC.
*
     ACCEPT WORK1 FROM TIME.
     DIVIDE 100 INTO WORK1.
     MOVE WORK1 TO WS-TIME-9.
     MOVE WS-DATE-YY TO WS-FORMAT-DATE-YY.
     MOVE WS-DATE-MM TO WS-FORMAT-DATE-MM.
     MOVE WS-DATE-DD TO WS-FORMAT-DATE-DD.
     ACCEPT WORK1 FROM TIME.
     DIVIDE 100 INTO WORK1.
     MOVE   WORK1 TO WS-FORMAT-TIME-SS.
     DIVIDE 100 INTO WORK1.
     MOVE   WORK1 TO WS-FORMAT-TIME-MM.
     DIVIDE 100 INTO WORK1.
     MOVE   WORK1 TO WS-FORMAT-TIME-HH.
     PERFORM  1200-SETUP-INPUT.
*--SET COMMON ERROR INFO
     MOVE  ZERO         TO  ERROR-CODE.
     MOVE 'CALLER'      TO  ERROR-PROGRAM
     MOVE  WS-FORMATTED-DATE  TO  ERR-FORMATTED-DATE.
     MOVE  WS-FORMATTED-TIME  TO  ERR-FORMATTED-TIME.
*
*-------------------------------------------------------------*
 1000-EXIT.
     EXIT.
  EJECT
*-------------------------------------------------------------*
 1200-SETUP-INPUT.
*-------------------------------------------------------------*
     ACCEPT LK-DATA FROM SYSIPT.
     MOVE LK-DATA TO WS-DATA-ALL.
     MOVE  WS-DATA-TIMES  TO WS-PROCESS-TIMES.
     IF WS-PROCESS-TIMES  EQUAL  ZERO
      THEN
           MOVE  100  TO   WS-PROCESS-TIMES.
     GO TO  1000-EXIT.
*-------------------------------------------------------------*
  EJECT
*-------------------------------------------------------------*
```

```
 2000-PUT-MESSAGES.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          PUT
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME-CONNECT.
     MOVE MQCC-OK                       TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE                     TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'EZBMBTCH'  USING  MQCONN
                          WS-QM-NAME-AREA
                          WS-HCONN-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-OUTPUT    TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES         TO   MQOD-OBJECTQMGRNAME.
     MOVE WS-DATA-QUEUE  TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK        TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE      TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE   TO   NULL.
     CALL 'EZBMBTCH'  USING  MQOPEN
                          WS-HCONN-ADDR-AREA
                          WS-Q-NAME-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-HOBJ-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*--CHECK IF MUST PUT TIME STAMP ON MESSAGE
       IF WS-PUT-TIMESTAMP
        THEN
             PERFORM 8000-GET-TIME-STAMP
             MOVE WS-TIMESTAMP-VALUE  TO  WS-BUFFER-TS
             MOVE  LENGTH OF WS-BUFFER-TS
                                  TO WS-BUFFER-LENGTH
             ADD WS-MSG-LENGTH    TO WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA        TO  WS-BUFFER-TEXT
        ELSE
             MOVE WS-MSG-LENGTH      TO  WS-BUFFER-LENGTH
             MOVE WS-MSG-AREA        TO  WS-BUFFER-AREA
       END-IF
*--MQPUT TO QUEUE TO QM
       MOVE  'PUT'     TO   WS-FUNCTION
```

```
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       CALL 'EZBMBTCH'  USING  MQPUT
                            WS-HCONN-ADDR-AREA
                            WS-HOBJ-ADDR-AREA
                            WS-MSG-DESCRIPTOR
                            WS-PUT-OPTIONS
                            WS-BUFFER-L-AREA
                            WS-BUFFER-AREA
                            WS-CCODE-ADDR-AREA
                            WS-RCODE-ADDR-AREA
*
       IF WS-CCODE-VALUE   NOT EQUAL  ZERO
        THEN
                GO TO  9900-ERR-DISPLAY
       END-IF
       ADD  +1         TO   WS-COUNT
     END-PERFORM.
*-----------------------------------------------------------------*
*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'EZBMBTCH'  USING  MQCLOSE
                          WS-HCONN-ADDR-AREA
                          WS-HOBJ-ADDR-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'EZBMBTCH'  USING  MQDISC
                          WS-HCONN-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*-----------------------------------------------------------------*
 2000-EXIT.
     EXIT.
  EJECT
*-----------------------------------------------------------------*
 3000-GET-MESSAGES.
*-----------------------------------------------------------------*
* PURPOSE: CONNECT , OPEN
*          GET
*          CLOSE,    DISCONNECT
*-----------------------------------------------------------------*
*
*--MQCONNECT TO QM
```

```
     MOVE  'CONNECT' TO   WS-FUNCTION.
     MOVE  SPACES    TO   WS-QM-NAME.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     SET WS-HCONN-VALUE  TO NULL.
     CALL 'EZBMBTCH'  USING  MQCONN
                          WS-QM-NAME-AREA
                          WS-HCONN-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*
*--MQOPEN   QUEUE TO QM
     MOVE  'OPEN'    TO   WS-FUNCTION.
     MOVE MQOO-INPUT-SHARED TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE SPACES            TO   MQOD-OBJECTQMGRNAME.
     MOVE WS-DATA-QUEUE     TO   MQOD-OBJECTNAME.
     MOVE MQCC-OK           TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE         TO   WS-RCODE-VALUE.
     SET WS-HOBJ-VALUE   TO   NULL.
     CALL 'EZBMBTCH'  USING  MQOPEN
                          WS-HCONN-ADDR-AREA
                          WS-Q-NAME-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-HOBJ-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*-----------------------------------------------------------------*
     PERFORM   WS-PROCESS-TIMES   TIMES
*
*--MQGET TO QUEUE TO QM
       MOVE  'GET'     TO   WS-FUNCTION
       MOVE MQCC-OK    TO   WS-CCODE-VALUE
       MOVE MQRC-NONE  TO   WS-RCODE-VALUE
       MOVE 500  TO   WS-BUFFER-LENGTH
       MOVE MQGMO-ACCEPT-TRUNCATED-MSG
                 TO   MQGMO-OPTIONS
*
       CALL 'EZBMBTCH'  USING  MQGET
                            WS-HCONN-ADDR-AREA
                            WS-HOBJ-ADDR-AREA
                            WS-MSG-DESCRIPTOR
                            WS-GET-OPTIONS
                            WS-BUFFER-L-AREA
                            WS-BUFFER-AREA
                            WS-DATA-L-AREA
                            WS-CCODE-ADDR-AREA
                            WS-RCODE-ADDR-AREA
*
       IF  (WS-CCODE-VALUE   NOT EQUAL ZERO)
        THEN
             IF WS-RCODE-VALUE EQUAL 2079
```

```
              THEN
                   SET WS-TRUNCATED-MESSAGES  TO TRUE
              ELSE
             IF WS-RCODE-VALUE EQUAL 2033
              THEN
                   SET WS-END-OF-MESSAGES  TO TRUE
                   GO TO     3000-GET-EOF
              ELSE
                  GO TO  9900-ERR-DISPLAY
             END-IF
             END-IF
*
       END-IF
       ADD  +1         TO   WS-COUNT

     END-PERFORM.
*-----------------------------------------------------------------*
*
 3000-GET-EOF.
*--MQCLOSE  QUEUE TO QM
     MOVE  'CLOSE'   TO   WS-FUNCTION.
     MOVE  ZERO      TO   WS-Q-OPEN-OPTIONS-VALUE.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'EZBMBTCH'  USING  MQCLOSE
                          WS-HCONN-ADDR-AREA
                          WS-HOBJ-ADDR-AREA
                          WS-Q-OPEN-OPTIONS
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.
*--MQDISC  FROM QM
     MOVE  'DISCONN' TO   WS-FUNCTION.
     MOVE MQCC-OK    TO   WS-CCODE-VALUE.
     MOVE MQRC-NONE  TO   WS-RCODE-VALUE.
     CALL 'EZBMBTCH'  USING  MQDISC
                          WS-HCONN-ADDR-AREA
                          WS-CCODE-ADDR-AREA
                          WS-RCODE-ADDR-AREA.
*
     IF WS-CCODE-VALUE   NOT EQUAL  ZERO
      THEN
                GO TO  9900-ERR-DISPLAY.

*-----------------------------------------------------------------*
 3000-EXIT.
     EXIT.
  EJECT
*-----------------------------------------------------------------*
*-----------------------------------------------------------------*
 7000-SEND-TOTALS.
*-----------------------------------------------------------------*
     ACCEPT WORK1 FROM TIME.
     SUBTRACT WORK1       FROM WORK2.
     DIVIDE 100 INTO WORK2.
     MOVE  WS-COUNT          TO   WS-OK-MESSAGES.
```

```
           MOVE  WS-DURATION-SECS  TO   WS-OK-TIME.
           MOVE  WS-DATA-QUEUE     TO   WS-OK-QUEUE.
      *-- --MOVE REST
           MOVE WS-OK-STATS-LINE-1 TO  WS-OK-MSG-1.
           MOVE WS-OK-STATS-LINE-3 TO  WS-OK-MSG-3.
           MOVE WS-OK-STATS-LINE-4 TO  WS-OK-MSG-4.
      *-- --CHECK IF ANY ERRORS
           IF WS-END-OF-MESSAGES
            THEN
                  MOVE 'NO MORE MESSAGES'    TO WS-OK-MSG-5.
      *
           IF WS-TRUNCATED-MESSAGES
            THEN
                  MOVE 'TRUNCATED MESSAGES'  TO WS-OK-MSG-6.
      *
           IF WS-ERR-MSG
            THEN
              DISPLAY WS-ALL-MSG UPON SYSLST
            ELSE
              DISPLAY WS-OK-MSG UPON SYSLST.
      *-----------------------------------------------------------------*
         EJECT
      *-----------------------------------------------------------------*
       8000-GET-TIME-STAMP.
      *-----------------------------------------------------------------*
           ACCEPT WS-DATE-YYMMDD FROM DATE.
           ACCEPT WORK1    FROM TIME.
           DIVIDE WORK1 BY 100 GIVING WS-TIME-9.
      *
           MOVE WS-DATE-YYMMDD TO   WS-TIMESTAMP-DATE.
           MOVE WS-TIME-HHMMSS TO   WS-TIMESTAMP-TIME.
      *-----------------------------------------------------------------*
         EJECT
      *-----------------------------------------------------------------*
       9900-ERR-DISPLAY.
      *-----------------------------------------------------------------*
           SET WS-ERR-MSG  TO TRUE.
           MOVE WS-CCODE-VALUE TO  WS-ERR-DISPLAY-CCODE.
           MOVE WS-RCODE-VALUE TO  WS-ERR-DISPLAY-RCODE.
      *
           DISPLAY WS-ERR-DISPLAY UPON SYSLST.
           GO TO  0000-SEND-TOTALS.
         EJECT
```

E.2.2 EZBMBTCH

This is the source code of the assembler subroutine for batch, which uses the XPCC interface to pass MQI requests to the online sub-task contained in EZBMTASK.

```
         PRINT ON,GEN
* ***************************************************************
*
*   PROGRAM-ID.  EZBMBTCH
*
*   USAGE.       IS LINKED INTO A BATCH PROGRAM THAT REQUIRES ACCESS
*                TO MQ QUEUES IN VSE. THE MQ CALLS ARE MODIFIED
*                TO HAVE EZBMBTCH AS THE CALLED PROGRAM, WITH
*                THE MQ SERVICE REQUIRED AS THE FIRST PARAMETER.
*                ALL OTHER PARAMETERS STAY EXACTLY AS THEY WOULD
```

```
*                IN THE CICS ENVIRONMENT.
*
* ***************************************************************
EZBMBTCH CSECT
         DFHREGS
* ***************************************************************
*   ESTABLISH ADDRESSABILITY
* ***************************************************************
         STM   14,12,12(13)            SAVE CALLERS REGISTERS
         USING *-4,3,6                 USE R3 AND R6 AS BASE REGS
         LR    3,15                    LOAD BASE REG
         LR    6,3                     LOAD SECONDARY BASE REG
         AH    R6,FOURK                ADD 4096
         LR    R10,R1                  SAVE R1 FOR LATER USE
         SPACE 3
         WTO   'IN SUBROUTINE'
         SPACE 3
         CLI   FIRSTUSE,C'S'           BEEN HERE BEFORE?
         BE    MOVEPARM                * YES, MOVE PARAMETERS
         MVI   FIRSTUSE,C'S'           SET FIRSTUSE PARM
         SPACE 3
* ********************************************************************
*  XPCC INITIALISATION STARTS
* ********************************************************************
         LA    R4,XPCCB1                POINT TO XPCCB
         USING IJBXPCCB,R4
         XPCC  XPCCB=(R4),FUNC=IDENT,   IDENTIFY                      X
               FDSCR=UNIQUE
         CH    R15,EIGHT                AM I UNIQUE?
         BNL   ERROR                    * NO .. ERROR
         SPACE 3
         WTO   'IDENTIFIED'
         SPACE 3
*        PRINT ON,GEN
         XPCC  XPCCB=(R4),FUNC=CONNECT, CONNECT                       X
               MECB=XPCCECB1
         LTR   R15,R15                  CONNECTED BOTH SIDES?
         BNZ   ERROR                    * NO .. ERROR IN CONNECTION
         SPACE 3
         WTO   'CONNECTED'
         SPACE 3
* ********************************************************************
*  MOVE PARAMETER LIST TO SAVE AREA
* ********************************************************************
MOVEPARM DS    0H
         LA    R7,SAVEAREA              ADDRESS OWN SAVE AREA
         LR    R8,R10                   INIT WORK REG
         LR    R9,R7                    INIT WORK REG
LOOP1    DS    0H
         MVC   0(4,R9),0(R8)            MOVE PARAMETER TO SAVE-AREA
         MVC   TEST,0(R8)               TEST FIRST CHARACTER OF PARAM
         NI    TEST,X'80'               * IS IT LAST PARAMETER?
         BNZ   LOOP1FIN                 *  NO, MOVE MORE DATA
         LA    R8,4(0,R8)               GET NEXT PARAM
         LA    R9,4(0,R9)               GET NEXT SAVE SLOT
         B     LOOP1                    * AND GO ROUND AGAIN
*
LOOP1FIN DS    0H
```

```
*
*
         LA    R5,BUFF1                ADDRESS BUFFER
         LR    R7,R10                  INITIALISE WORK REG
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R5),0(R8)           MOVE FIRST PARAM INTO BUFFER
* ********************************************************************
*  CHECK WHAT MQ SERVICE IS REQUIRED AND HANDLE IT
* ********************************************************************
         SPACE 3
         CLC   0(8,R5),=CL8'EZDISC '
         BE    XPCCWORK
         SPACE 3
         CLC   0(8,R5),=CL8'MQCLOSE'
         BE    PMQCLOSE
         SPACE 3
         CLC   0(8,R5),=CL8'MQCONN'
         BE    PMQCONN
         SPACE 3
         CLC   0(8,R5),=CL8'MQDISC'
         BE    PMQDISC
         SPACE 3
         CLC   0(8,R5),=CL8'MQGET'
         BE    PMQGET
         SPACE 3
         CLC   0(8,R5),=CL8'MQOPEN'
         BE    PMQOPEN
         SPACE 3
         CLC   0(8,R5),=CL8'MQPUT'
         BE    PMQPUT
         SPACE 3
*
* ********************************************************************
*  STANDARD ERROR HANDLER HERE
* ********************************************************************
         WTO   'NO PROPER REQUEST ENTERED'
         B     ERROR
* ********************************************************************
*  STANDARD ERROR HANDLER HERE
* ********************************************************************
*
PMQCLOSE DS    0H
         WTO   'IN MQCLOSE'
         LA    R5,8(0,R5)              POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW HOBJ
         LA    R7,4(0,R7)              POINT AT OLD HOBJ ADDRESS
         L     R8,0(0,R7)              LOAD HOBJ ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW OPTIONS
         LA    R7,4(0,R7)              POINT AT OLD OPTIONS ADDRESS
         L     R8,0(0,R7)              LOAD OPTIONS ADDRESS
         MVC   0(4,R5),0(R8)           LOAD OPTIONS INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
```

```
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R5),0(R8)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R5),0(R8)           LOAD REASON INTO BUFFER
*
         BAL   R9,XPCCWORK             DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                ADDRESS BUFFER
*        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
         LR    R7,R10                  INITIALIZE WORK REG
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)           MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)              POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)           MOVE HCONN OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW HOBJ
         LA    R7,4(0,R7)              POINT AT OLD HOBJ ADDRESS
         L     R8,0(0,R7)              LOAD HOBJ ADDRESS
         MVC   0(4,R8),0(R5)           MOVE HOBJ OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW OPTIONS
         LA    R7,4(0,R7)              POINT AT OLD OPTIONS ADDRESS
         L     R8,0(0,R7)              LOAD OPTIONS ADDRESS
         MVC   0(4,R8),0(R5)           MOVE OPTIONS OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)           MOVE COMPCODE OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R8),0(R5)           MOVE REASON OUT OF BUFFER
*
         B     END
*
PMQCONN  DS    0H
         WTO   'IN MQCONN'
         LA    R5,8(0,R5)              POINT AT NEW NAME
         LA    R7,4(0,R7)              POINT AT OLD NAME ADDRESS
         L     R8,0(0,R7)              LOAD NAME ADDRESS
         MVC   0(48,R5),0(R8)          LOAD NAME INTO BUFFER
*
         LA    R5,48(0,R5)             POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
```

```
         MVC   0(4,R5),0(R8)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R5),0(R8)           LOAD REASON INTO BUFFER
*
         BAL   R9,XPCCWORK             DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                ADDRESS BUFFER
*        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
         LR    R7,R10                  INITIALIZE WORK REG
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)           MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)              POINT AT OLD NAME
         LA    R7,4(0,R7)              POINT AT NEW NAME ADDRESS
         L     R8,0(0,R7)              LOAD NAME ADDRESS
         MVC   0(48,R8),0(R5)          LOAD NAME INTO BUFFER
*
         LA    R5,48(0,R5)             POINT AT OLD HCONN
         LA    R7,4(0,R7)              POINT AT NEW HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD COMPCODE
         LA    R7,4(0,R7)              POINT AT NEW COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD REASON
         LA    R7,4(0,R7)              POINT AT NEW REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R8),0(R5)           LOAD REASON INTO BUFFER
*
         B     END
*
PMQDISC  DS    0H
         WTO   'IN MQDISC'
         LA    R5,8(0,R5)              POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R5),0(R8)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASONS ADDRESS
         L     R8,0(0,R7)              LOAD REASONS ADDRESS
         MVC   0(4,R5),0(R8)           LOAD REASONS INTO BUFFER
*
         BAL   R9,XPCCWORK             DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                ADDRESS BUFFER
*        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
         LR    R7,R10                  INITIALIZE WORK REG
```

```
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)           MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)              POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)           MOVE HCONN OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)           MOVE COMPCODE OUT OF BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASONS
         LA    R7,4(0,R7)              POINT AT OLD REASONS ADDRESS
         L     R8,0(0,R7)              LOAD REASONS ADDRESS
         MVC   0(4,R8),0(R5)           MOVE REASONS OUT OF BUFFER
*
         B     END
*
PMQGET   DS    0H
         WTO   'IN MQGET'
         LA    R5,8(0,R5)              POINT AT NEW HCONN
         LA    R7,4(0,R7)              POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW HOBJ
         LA    R7,4(0,R7)              POINT AT OLD HOBJ ADDRESS
         L     R8,0(0,R7)              LOAD HOBJ ADDRESS
         MVC   0(4,R5),0(R8)           LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW MSGDESC
         LA    R7,4(0,R7)              POINT AT OLD MSGDESC ADDRESS
         L     R8,0(0,R7)              LOAD MSGDESC ADDRESS
         LA    R9,324
L1       DS    0H
         MVC   0(1,R5),0(R8)           LOAD A CHARACTER
         LA    R5,1(0,R5)              INCREMENT NEW BUFFER POINTER
         LA    R8,1(0,R8)              INCREMENT OLD BUFFER POINTER
         BCT   R9,L1                   LOOP ROUND AND MOVE NEXT CHAR
*
         LA    R7,4(0,R7)              POINT AT OLD GETSOPTS ADDRESS
         L     R8,0(0,R7)              LOAD GETSOPTS ADDRESS
         MVC   0(72,R5),0(R8)          LOAD GETSOPTS INTO BUFFER
*
         LA    R5,72(0,R5)             POINT AT NEW BUFFER-LEN
         LA    R7,4(0,R7)              POINT AT OLD BUFFER-LEN ADDRESS
         L     R8,0(0,R7)              LOAD BUFFER-LEN ADDRESS
         MVC   0(4,R5),0(R8)           LOAD BUFFER-LEN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW BUFFER
         LA    R7,4(0,R7)              POINT AT OLD BUFFER ADDRESS
         L     R8,0(0,R7)              LOAD BUFFER ADDRESS
         LA    R9,516                  MAXIMUM BUFFER LENGTH
*              CHANGE THE LINE ABOVE IF YOU WANT BIGGER BUFFERS
L2       DS    0H
         MVC   0(1,R5),0(R8)           LOAD A CHARACTER
         LA    R5,1(0,R5)              INCREMENT NEW BUFFER POINTER
```

```
         LA    R8,1(0,R8)              INCREMENT OLD BUFFER POINTER
         BCT   R9,L2                   LOOP ROUND AND MOVE NEXT CHAR
*
*        LA    R5,4(0,R5)              POINT AT NEW DATALENG
         LA    R7,4(0,R7)              POINT AT OLD DATALENG ADDRESS
         L     R8,0(0,R7)              LOAD DATALENG ADDRESS
         MVC   0(4,R5),0(R8)           LOAD DATALENG INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R5),0(R8)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R5),0(R8)           LOAD REASON INTO BUFFER
*
         BAL   R9,XPCCWORK             DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                ADDRESS BUFFER
*        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
         LR    R7,R10                  INITIALIZE WORK REG
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)           MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)              POINT AT OLD HCONN
         LA    R7,4(0,R7)              POINT AT NEW HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD HOBJ
         LA    R7,4(0,R7)              POINT AT NEW HOBJ ADDRESS
         L     R8,0(0,R7)              LOAD HOBJ ADDRESS
         MVC   0(4,R8),0(R5)           LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD MSGDESC
         LA    R7,4(0,R7)              POINT AT NEW MSGDESC ADDRESS
         L     R8,0(0,R7)              LOAD MSGDESC ADDRESS
         LA    R9,324
L3       DS    0H
         MVC   0(1,R5),0(R8)           LOAD A CHARACTER
         LA    R5,1(0,R5)              INCREMENT OLD BUFFER POINTER
         LA    R8,1(0,R8)              INCREMENT NEW BUFFER POINTER
         BCT   R9,L3                   LOOP ROUND AND MOVE NEXT CHAR
*
         LA    R7,4(0,R7)              POINT AT NEW GETSOPTS ADDRESS
         L     R8,0(0,R7)              LOAD GETSOPTS ADDRESS
         MVC   0(72,R8),0(R5)          LOAD GETSOPTS INTO BUFFER
*
         LA    R5,72(0,R5)             POINT AT OLD BUFFER-LEN
         LA    R7,4(0,R7)              POINT AT NEW BUFFER-LEN ADDRESS
         L     R8,0(0,R7)              LOAD BUFFER-LEN ADDRESS
         MVC   0(4,R8),0(R5)           LOAD BUFFER-LEN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD BUFFER
         LA    R7,4(0,R7)              POINT AT NEW BUFFER ADDRESS
         L     R8,0(0,R7)              LOAD BUFFER ADDRESS
```

```
         LA    R9,516                   MAXIMUM BUFFER LENGTH
*              CHANGE THE LINE ABOVE IF YOU WANT BIGGER BUFFERS
L4       DS    0H
         MVC   0(1,R8),0(R5)            LOAD A CHARACTER
         LA    R5,1(0,R5)               INCREMENT OLD BUFFER POINTER
         LA    R8,1(0,R8)               INCREMENT NEW BUFFER POINTER
         BCT   R9,L4                    LOOP ROUND AND MOVE NEXT CHAR
*
*        LA    R5,4(0,R5)               POINT AT OLD DATALENG
         LA    R7,4(0,R7)               POINT AT NEW DATALENG ADDRESS
         L     R8,0(0,R7)               LOAD DATALENG ADDRESS
         MVC   0(4,R8),0(R5)            LOAD DATALENG INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD COMPCODE
         LA    R7,4(0,R7)               POINT AT NEW COMPCODE ADDRESS
         L     R8,0(0,R7)               LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)            LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD REASON
         LA    R7,4(0,R7)               POINT AT NEW REASON ADDRESS
         L     R8,0(0,R7)               LOAD REASON ADDRESS
         MVC   0(4,R8),0(R5)            LOAD REASON INTO BUFFER
*
         B     END
*
*
PMQOPEN  DS    0H
         WTO   'IN MQOPEN'
         LA    R5,8(0,R5)               POINT AT NEW HCONN
         LA    R7,4(0,R7)               POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)               LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)            LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW OBJDESC
         LA    R7,4(0,R7)               POINT AT OLD OBJDESC ADDRESS
         L     R8,0(0,R7)               LOAD OBJDESC ADDRESS
         MVC   0(168,R5),0(R8)          LOAD OBJDESC INTO BUFFER
*
         LA    R5,168(0,R5)             POINT AT NEW OPTIONS
         LA    R7,4(0,R7)               POINT AT OLD OPTIONS ADDRESS
         L     R8,0(0,R7)               LOAD OPTIONS ADDRESS
         MVC   0(4,R5),0(R8)            LOAD OPTIONS INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW HOBJ
         LA    R7,4(0,R7)               POINT AT OLD HOBJ ADDRESS
         L     R8,0(0,R7)               LOAD HOBJ ADDRESS
         MVC   0(4,R5),0(R8)            LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW COMPCODE
         LA    R7,4(0,R7)               POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)               LOAD COMPCODE ADDRESS
         MVC   0(4,R5),0(R8)            LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW REASON
         LA    R7,4(0,R7)               POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)               LOAD REASON ADDRESS
         MVC   0(4,R5),0(R8)            LOAD REASON INTO BUFFER
*
```

```
         BAL   R9,XPCCWORK              DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                 ADDRESS BUFFER
*        LA    R5,100(0,R5)             POINT AT PARAMETER AREA
         LR    R7,R10                   INITIALIZE WORK REG
         L     R8,0(0,R7)               LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)            MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)               POINT AT OLD HCONN
         LA    R7,4(0,R7)               POINT AT NEW HCONN ADDRESS
         L     R8,0(0,R7)               LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)            LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD OBJDESC
         LA    R7,4(0,R7)               POINT AT NEW OBJDESC ADDRESS
         L     R8,0(0,R7)               LOAD OBJDESC ADDRESS
         MVC   0(168,R8),0(R5)          LOAD OBJDESC INTO BUFFER
*
         LA    R5,168(0,R5)             POINT AT OLD OPTIONS
         LA    R7,4(0,R7)               POINT AT NEW OPTIONS ADDRESS
         L     R8,0(0,R7)               LOAD OPTIONS ADDRESS
         MVC   0(4,R8),0(R5)            LOAD OPTIONS INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD HOBJ
         LA    R7,4(0,R7)               POINT AT NEW HOBJ ADDRESS
         L     R8,0(0,R7)               LOAD HOBJ ADDRESS
         MVC   0(4,R8),0(R5)            LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD COMPCODE
         LA    R7,4(0,R7)               POINT AT NEW COMPCODE ADDRESS
         L     R8,0(0,R7)               LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)            LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD REASON
         LA    R7,4(0,R7)               POINT AT NEW REASON ADDRESS
         L     R8,0(0,R7)               LOAD REASON ADDRESS
         MVC   0(4,R8),0(R5)            LOAD REASON INTO BUFFER
*
         B     END
*
PMQPUT   DS    0H
         WTO   'IN MQPUT'
         LA    R5,8(0,R5)               POINT AT NEW HCONN
         LA    R7,4(0,R7)               POINT AT OLD HCONN ADDRESS
         L     R8,0(0,R7)               LOAD HCONN ADDRESS
         MVC   0(4,R5),0(R8)            LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW HOBJ
         LA    R7,4(0,R7)               POINT AT OLD HOBJ ADDRESS
         L     R8,0(0,R7)               LOAD HOBJ ADDRESS
         MVC   0(4,R5),0(R8)            LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT NEW MSGDESC
         LA    R7,4(0,R7)               POINT AT OLD MSGDESC ADDRESS
         L     R8,0(0,R7)               LOAD MSGDESC ADDRESS
         LA    R9,324                   LOAD LOOP COUNTER
LB       DS    0H
         MVC   0(1,R5),0(R8)            LOAD A CHARACTER
         LA    R5,1(0,R5)               INCREMENT OLD BUFFER POINTER
```

```
         LA    R8,1(0,R8)              INCREMENT NEW BUFFER POINTER
         BCT   R9,LB                   LOOP ROUND AND MOVE NEXT CHAR
*
         LA    R7,4(0,R7)              POINT AT OLD PUTMSGOPT ADDRESS
         L     R8,0(0,R7)              LOAD PUTMSHOPT ADDRESS
         MVC   0(128,R5),0(R8)         LOAD PUTMSGOPT INTO BUFFER
*
         LA    R5,128(0,R5)            POINT AT NEW BUFFLENG
         LA    R7,4(0,R7)              POINT AT OLD BUFFLENG ADDRESS
         L     R8,0(0,R7)              LOAD BUFFLENG ADDRESS
         MVC   0(4,R5),0(R8)           LOAD BUFFLENG INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW BUFFER
         LA    R7,4(0,R7)              POINT AT OLD BUFFER ADDRESS
         L     R8,0(0,R7)              LOAD BUFFER ADDRESS
         LA    R9,516                  MAXIMUM BUFFER LENGTH
*              CHANGE THE LINE ABOVE IF YOU WANT BIGGER BUFFERS
LC       DS    0H
         MVC   0(1,R5),0(R8)           LOAD A CHARACTER
         LA    R5,1(0,R5)              INCREMENT NEW BUFFER POINTER
         LA    R8,1(0,R8)              INCREMENT OLD BUFFER POINTER
         BCT   R9,LC                   LOOP ROUND AND MOVE NEXT CHAR
*
*        LA    R5,4(0,R5)              POINT AT NEW COMPCODE
         LA    R7,4(0,R7)              POINT AT OLD COMPCODE ADDRESS
         L     R8,0(0,R7)              LOAD COMPCODE ADDRESS
         MVC   0(4,R5),0(R8)           LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT NEW REASON
         LA    R7,4(0,R7)              POINT AT OLD REASON ADDRESS
         L     R8,0(0,R7)              LOAD REASON ADDRESS
         MVC   0(4,R5),0(R8)           LOAD REASON INTO BUFFER
*
         BAL   R9,XPCCWORK             DO THE XPCC SEND/RECEIVE
*   NOW MOVE ALL THE DATA FROM THE BUFFER
         LA    R5,BUFF1                ADDRESS BUFFER
*        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
         LR    R7,R10                  INITIALIZE WORK REG
         L     R8,0(0,R7)              LOAD ADDRESS OF 1ST PARAM
         MVC   0(8,R8),0(R5)           MOVE FIRST PARAM OUT OF BUFFER
*
         LA    R5,8(0,R5)              POINT AT OLD HCONN
         LA    R7,4(0,R7)              POINT AT NEW HCONN ADDRESS
         L     R8,0(0,R7)              LOAD HCONN ADDRESS
         MVC   0(4,R8),0(R5)           LOAD HCONN INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD HOBJ
         LA    R7,4(0,R7)              POINT AT NEW HOBJ ADDRESS
         L     R8,0(0,R7)              LOAD HOBJ ADDRESS
         MVC   0(4,R8),0(R5)           LOAD HOBJ INTO BUFFER
*
         LA    R5,4(0,R5)              POINT AT OLD MSGDESC
         LA    R7,4(0,R7)              POINT AT NEW MSGDESC ADDRES
         L     R8,0(0,R7)              LOAD MSGDESC ADDRESS
         LA    R9,324                  LOAD LOOP COUNTER
LD       DS    0H
         MVC   0(1,R8),0(R5)           LOAD A CHARACTER
         LA    R5,1(0,R5)              INCREMENT OLD BUFFER POINTER
         LA    R8,1(0,R8)              INCREMENT NEW BUFFER POINTER
```

```
         BCT   R9,LD                    LOOP ROUND AND MOVE NEXT CHAR
*
         LA    R7,4(0,R7)               POINT AT NEW PUTMSGOPT ADDRESS
         L     R8,0(0,R7)               LOAD PUTMSHOPT ADDRESS
         MVC   0(128,R8),0(R5)          LOAD PUTMSGOPT INTO BUFFER
*
         LA    R5,128(0,R5)             POINT AT OLD BUFFLENG
         LA    R7,4(0,R7)               POINT AT NEW BUFFLENG ADDRESS
         L     R8,0(0,R7)               LOAD BUFFLENG ADDRESS
         MVC   0(4,R8),0(R5)            LOAD BUFFLENG INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD BUFFER
         LA    R7,4(0,R7)               POINT AT NEW BUFFER ADDRESS
         L     R8,0(0,R7)               LOAD BUFFER ADDRESS
         LA    R9,516                   MAXIMUM BUFFER LENGTH
*               CHANGE THE LINE ABOVE IF YOU WANT BIGGER BUFFERS
LE       DS    0H
         MVC   0(1,R8),0(R5)            LOAD A CHARACTER
         LA    R5,1(0,R5)               INCREMENT OLD BUFFER POINTER
         LA    R8,1(0,R8)               INCREMENT NEW BUFFER POINTER
         BCT   R9,LE                    LOOP ROUND AND MOVE NEXT CHAR
*
         LA    R7,4(0,R7)               POINT AT NEW COMPCODE ADDRESS
         L     R8,0(0,R7)               LOAD COMPCODE ADDRESS
         MVC   0(4,R8),0(R5)            LOAD COMPCODE INTO BUFFER
*
         LA    R5,4(0,R5)               POINT AT OLD REASON
         LA    R7,4(0,R7)               POINT AT NEW REASON ADDRESS
         L     R8,0(0,R7)               LOAD REASON ADDRESS
         MVC   0(4,R8),0(R5)            LOAD REASON INTO BUFFER
*
*
         B     END
*
*
*
*
*
* ********************************************************************
*  XPCC SEND/RECEIVE
* ********************************************************************
XPCCWORK DS    0H
         LA    R4,XPCCB1                POINT TO XPCCB
         WTO   'IN XPCCWORK'
         XPCC  XPCCB=(R4),              SEND DATA BACK                X
               FUNC=SEND
         LA    R2,IJBXSECB
         WAIT  (R2)                     SEND COMPLETE
         MVC   IJBXSECB,ZEROF
         LTR   R15,R15
         BNZ   SENDERR
         SPACE 3
         WTO   'SENT'
         CLC   0(8,R5),=CL8'EZDISC '
         BE    DISCONCT
         LA    R2,IJBXRECB
         WAIT  (R2)                     WAIT FOR A MESSAGE
         MVC   IJBXRECB,ZEROF
         XPCC  XPCCB=(R4),              RECEIVE REQUEST               X
```

```
              FUNC=RECEIVE
         LTR   R15,R15
         BNZ   RECERR
         SPACE 3
         WTO   'RECEIVED'
         BR    R9                        RETURN TO CALLER
* ********************************************************************
*  XPCC DISCONNECT ETC
* ********************************************************************
DISCONCT DS    OH
         XPCC  XPCCB=(R4),               TERMINATE CONNECTION          X
               FUNC=DISCONN
         WTO   'DISCONNECTED'
         XPCC  XPCCB=(R4),               TERMINATE XPCC                X
               FUNC=TERMIN
         WTO   'TERMINATED'
*
*
*
*
END      DS    OH
         SR    R15,R15                 SET RETURN CODE
*
ERROR    DS    OH
         LM    14,12,12(13)            RESTORE CALLERS REGS
         BR    R14                     RETURN
SENDERR  DS    OH
         WTO   'ERROR IN SEND'
         B     ERROR
RECERR   DS    OH
         WTO   'ERROR IN RECEIVE'
         B     ERROR
*
*
*        PRINT ON,GEN
XPCCB1   XPCCB APPL=EZBAPPLB,TOAPPL=EZBAPPLC,                          X
               BUFFER=(BUFF1,L'BUFF1),VERSION=2
XPCCECB1 TECB
EIGHT    DC    H'08'
FOUR     DC    H'04'
FOURK    DC    H'4096'
FOURF    DC    F'04'
BUFFLENG DS    F
SAVEAREA DS    25F
TEST     DS    CL1
FIRSTUSE DC    C' '
ZEROF    DC    F'00'
*
*
*
         DS    OF
BUFF1    DS    CL4096
         MAPXPCCB VERSION=2
*
         END
```

E.2.3 Job Control to Run the Batch Interface

```
* $$ JOB JNM=CALLER,DISP=L,CLASS=A
* $$ LST DISP=D,CLASS=A,PRI=5
// JOB CALLER
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(MQMUSR1.RICH,PRD2.DBASE,PRD2.CICSR)
// EXEC CALLER,SIZE=512K
PUT  02 XMIT8
/*
/&
* $$ EOJ
```

E.3 MQICICS Online COBOL Program

This is a sample COBOL II program to execute functions received from the batch interface program. The individual calls are based on those issued in TTPTST2, but the call selection logic is specific to this sample.

This program is invoked by an EXEC CICS LINK request from the assembly language program EZBMTASK. It is passed a COMMAREA, which contains an eight-byte function code in the first eight bytes. The MQI is invoked with the parameters pointing to the appropriate offsets in the COMMARREA, and consequently no reformatting is required on return. MQICICS does not test any return code, and will return all return codes to the batch program (CALLER), so CALLER must take responsibility for error checking and recovery.

E.3.1 MQICICS Source Statements

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    MQICICS.
      * RECEIVE PARM FROM XPCCPGM
      ****************************************************************
      *               T E S T                                        *
      *                                                              *
      *          A P P L I C A T I O N   I N T E R F A C E           *
      *                                                              *
      *                 IBM MQI                                      *
      *                                                              *
      *--------------------------------------------------------------*
      * MQICICS - MQI BATCH INTERFACE TESTE PROGRAM ONLINE COMPONENT   *
      *                                                              *
      * FUNCTIONS:  1. PERFORM NORMAL QUEUE PUT,GET, and PUT1        *
      *                                                              *
      * COPYBOOKS:  MQIVALUE - MQI RETURN CODES.                     *
      *                                                              *
      * CALLS    :  MQCONN   - CONNECT                               *
      *             MQOPEN   - OPEN                                  *
      *             MQPUT    - PUT                                   *
      *             MQPUT1   - PUT1                                  *
      *             MQINQ    - INQUIRY                               *
      *             MQGET    - GET                                   *
      *             MQCLOSE  - CLOSE                                 *
      *             MQDISC   - DISCONNECT                            *
      *                                                              *
      * CALLED BY:  -- NONE --                                       *
      *                                                              *
      * CHANGE SUMMARY:                                              *
```

```
*                                                               *
*---------------------------------------------------------------*
/
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  FILLER                          PIC X(40) VALUE
                 'MQICICS WORKING STORAGE STARTS HERE ==>'.
 77  PARM-LENGTH           PIC S9(4)  COMP VALUE ZEROS.
 01  PARM-RECEIVED.
     05  PARM-FUNCTION.
        10  FUNCTION-RECEIVED   PIC XXXXXXXX  VALUE 'FUNCTION'.
           88  FUNCTION-CLOSE                  VALUE 'MQCLOSE'.
           88  FUNCTION-CONN                   VALUE 'MQCONN'.
           88  FUNCTION-DISC                   VALUE 'MQDISC'.
           88  FUNCTION-GET                    VALUE 'MQGET'.
           88  FUNCTION-INQ                    VALUE 'MQINQ'.
           88  FUNCTION-OPEN                   VALUE 'MQOPEN'.
           88  FUNCTION-PUT                    VALUE 'MQPUT'.
           88  FUNCTION-PUT1                   VALUE 'MQPUT1'.
        10  PARM-PROCESS.
           20  PARM-AREA               PIC X(4088) VALUE SPACES.
        10  CONN-PROCESS  REDEFINES PARM-PROCESS.
           20  CONN-QM-NAME-AREA       PIC X(48).
           20  CONN-HCONN-ADDR-AREA.
              25 CONN-HCONN-VALUE      USAGE POINTER.
           20  CONN-CCODE-ADDR-AREA.
             25 CONN-CCODE-VALUE       PIC S9(8) COMP.
           20  CONN-RCODE-ADDR-AREA.
             25 CONN-RCODE-VALUE       PIC S9(8) COMP.
           20  FILLER                  PIC X(4028).
        10  OPEN-PROCESS  REDEFINES PARM-PROCESS.
           20  OPEN-HCONN-ADDR-AREA.
              25 OPEN-HCONN-VALUE      USAGE POINTER.
           20  OPEN-Q-NAME-AREA        PIC X(168).
           20  OPEN-Q-OPEN-OPTIONS     PIC S9(9)  BINARY.
           20  OPEN-HOBJ-ADDR-AREA.
              25 OPEN-HOBJ-VALUE       USAGE POINTER.
           20  OPEN-CCODE-ADDR-AREA.
             25 OPEN-CCODE-VALUE       PIC S9(8) COMP.
           20  OPEN-RCODE-ADDR-AREA.
             25 OPEN-RCODE-VALUE       PIC S9(8) COMP.
           20  FILLER                  PIC X(3900).
        10  PUT-PROCESS   REDEFINES PARM-PROCESS.
           20  PUT-HCONN-ADDR-AREA.
              25 PUT-HCONN-VALUE       USAGE POINTER.
           20  PUT-HOBJ-ADDR-AREA.
              25 PUT-HOBJ-VALUE        USAGE POINTER.
           20  PUT-MSG-DESCRIPTOR      PIC X(324).
           20  PUT-PUT-OPTIONS         PIC X(128).
           20  PUT-BUFFER-L-AREA.
              25 PUT-BUFFER-LENGTH     PIC S9(8) COMP.
           20  PUT-BUFFER-AREA.
              25 PUT-BUFFER-TS         PIC X(16).
              25 PUT-BUFFER-TEXT       PIC X(500).
           20  PUT-CCODE-ADDR-AREA.
              25 PUT-CCODE-VALUE       PIC S9(8) COMP.
           20  PUT-RCODE-ADDR-AREA.
```

```
      25  PUT-RCODE-VALUE      PIC S9(8) COMP.
   20  FILLER                  PIC X(3100).
10  PUT1-PROCESS  REDEFINES PARM-PROCESS.
   20  PUT1-HCONN-ADDR-AREA.
      25 PUT1-HCONN-VALUE       USAGE POINTER.
   20  PUT1-Q-NQME-AREA        PIC X(168).
   20  PUT1-MSG-DESCRIPTOR     PIC X(324).
   20  PUT1-PUT-OPTIONS        PIC X(128).
   20  PUT1-BUFFER-L-AREA.
      25 PUT1-BUFFER-LENGTH    PIC S9(8) COMP.
   20  PUT1-BUFFER-AREA.
      25 PUT1-BUFFER-TS        PIC X(16).
      25 PUT1-BUFFER-TEXT      PIC X(500).
   20  PUT1-CCODE-ADDR-AREA.
      25  PUT1-CCODE-VALUE     PIC S9(8) COMP.
   20  PUT1-RCODE-ADDR-AREA.
      25  PUT1-RCCODE-VALUE    PIC S9(8) COMP.
   20  FILLER                  PIC X(2936).
10  INQ-PROCESS   REDEFINES PARM-PROCESS.
   20  INQ-HCONN-ADDR-AREA.
      25 INQ-HCONN-VALUE         USAGE POINTER.
   20  INQ-HOBJ-ADDR-AREA.
      25 INQ-HOBJ-VALUE          USAGE POINTER.
   20  INQ-MQI-SECTOR-COUNT      PIC S9(9)  BINARY.
   20  INQ-MQI-SECTOR            PIC S9(9)  BINARY.
   20  INQ-MQI-IN-ATTR-COUNT     PIC S9(9)  BINARY.
   20  INQ-MQI-IN-ATTR           PIC S9(9)  BINARY.
   20  INQ-MQI-CHAR-ATTR-LENGTH  PIC S9(9)  BINARY.
   20  INQ-MQI-CHAR-ATTR         PIC X(500).
   20  INQ-CCODE-ADDR-AREA.
      25 INQ-CCODE-VALUE         PIC S9(8) COMP.
   20  INQ-RCODE-ADDR-AREA.
      25 INQ-RCODE-VALUE         PIC S9(8) COMP.
   20  FILLER                    PIC X(3552).
10  GET-PROCESS   REDEFINES PARM-PROCESS.
   20  GET-HCONN-ADDR-AREA.
      25 GET-HCONN-VALUE           USAGE POINTER.
   20  GET-HOBJ-ADDR-AREA.
      25 GET-HOBJ-VALUE            USAGE POINTER.
   20  GET-MSG-DESCRIPTOR          PIC X(324).
   20  GET-GET-OPTIONS             PIC X(72).
   20  GET-BUFFER-L-AREA.
      25 GET-BUFFER-LENGTH    PIC S9(8) COMP.
   20  GET-BUFFER-AREA.
      25 GET-BUFFER-TS        PIC X(16).
      25 GET-BUFFER-TEXT      PIC X(500).
   20  GET-DATA-L-AREA.
      25 GET-DATA-LENGTH-USER PIC S9(8) COMP.
   20  GET-CCODE-ADDR-AREA.
      25 GET-CCODE-VALUE           PIC S9(8) COMP.
   20  GET-RCODE-ADDR-AREA.
      25 GET-RCODE-VALUE           PIC S9(8) COMP.
   20  FILLER                      PIC X(3152).
10  CLOSE-PROCESS   REDEFINES PARM-PROCESS.
   20  CLOSE-HCONN-ADDR-AREA.
      25 CLOSE-HCONN-VALUE         USAGE POINTER.
   20  CLOSE-HOBJ-ADDR-AREA.
      25 CLOSE-HOBJ-VALUE          USAGE POINTER.
   20  CLOSE-Q-OPEN-OPTIONS        PIC S9(9)  BINARY.
```

```
          20  CLOSE-CCODE-ADDR-AREA.
             25  CLOSE-CCODE-VALUE        PIC S9(8) COMP.
          20  CLOSE-RCODE-ADDR-AREA.
             25  CLOSE-RCODE-VALUE        PIC S9(8) COMP.
          20  FILLER                       PIC X(4068).
       10  DISC-PROCESS   REDEFINES PARM-PROCESS.
          20  DISC-HCONN-ADDR-AREA.
             25 DISC-HCONN-VALUE          USAGE POINTER.
          20  DISC-CCODE-ADDR-AREA.
             25  DISC-CCODE-VALUE         PIC S9(8) COMP.
          20  DISC-RCODE-ADDR-AREA.
             25  DISC-RCODE-VALUE         PIC S9(8) COMP.
          20 FILLER                       PIC X(4076).
 LINKAGE SECTION.
*-----------------------------------------------------------------*
 01  DFHCOMMAREA.

     05  PARM-DATA                PIC X(4096).
  EJECT
*-----------------------------------------------------------------*
 PROCEDURE DIVISION.
*-----------------------------------------------------------------*
 0000-MAIN-LINE.
*--INITIALIZE
     PERFORM 1000-INITIALIZE
        THRU 1000-EXIT.
*-----------------------------------------------------------------*
*   MQI FUNCTIONS.
*   CONNECT QUEUE
          IF FUNCTION-CONN
                  THEN   PERFORM   2000-CONNECT
                         THRU      2000-CONNECT-EXIT
                  GO TO 0000-RETURN
              END-IF
*   OPEN QUEUE
          IF FUNCTION-OPEN
                  THEN   PERFORM   2100-OPEN
                         THRU      2100-OPEN-EXIT
                  GO TO 0000-RETURN
              END-IF
*  PUT QUEUE RECORDS
          IF FUNCTION-PUT
                  THEN   PERFORM   2200-PUT
                         THRU      2200-PUT-EXIT
                  GO TO 0000-RETURN
              END-IF
*  GET  QUEUE RECORDS
          IF FUNCTION-GET
                 THEN   PERFORM   2300-GET
                        THRU      2300-GET-EXIT
                  GO TO 0000-RETURN
              END-IF
*  PUT1 QUEUE RECORDS
          IF FUNCTION-PUT1
                  THEN  PERFORM   2400-PUT1
                        THRU      2400-PUT1-EXIT
                  GO TO 0000-RETURN
              END-IF
*  INQUIRE QUEUE
```

```
          IF FUNCTION-INQ
                  THEN  PERFORM   2500-INQUIRE
                        THRU      2500-INQUIRE-EXIT
                  GO TO 0000-RETURN
              END-IF
*  CLOSE QUEUE
          IF FUNCTION-CLOSE
                  THEN  PERFORM   2600-CLOSE
                        THRU      2600-CLOSE-EXIT
                  GO TO 0000-RETURN
              END-IF
*  DISCONNECT QUEUE
          IF FUNCTION-DISC
                  THEN   PERFORM  2700-DISCONNECT
                         THRU     2700-DISCONNECT-EXIT
                  GO TO 0000-RETURN
              END-IF.
*    EXEC CICS ENTER TRACEID(90) END-EXEC.
         GO TO 0000-RETURN.
* ***************
 XXXX-ERROR.
*    EXEC CICS ENTER TRACEID(99) END-EXEC.
*---------------------------------------------------------------*
 0000-RETURN.
*---------------------------------------------------------------*
     EXEC CICS RETURN
          END-EXEC.
*---------------------------------------------------------------*
 1000-INITIALIZE.
*---------------------------------------------------------------*
* PURPOSE: SETUP DATA AREAS
*---------------------------------------------------------------*
      IF    EIBCALEN = 0
                     GO  TO    9900-NO-COMM-AREA
             ELSE
      MOVE          4096   TO    PARM-LENGTH.
      MOVE  DFHCOMMAREA TO PARM-RECEIVED.
*---------------------------------------------------------------*
 1000-EXIT.
    EXIT.
   EJECT
*---------------------------------------------------------------*
*---------------------------------------------------------------*
 2000-CONNECT.
*---------------------------------------------------------------*
*  CONNECT TO QM
*    EXEC CICS ENTER TRACEID(50) FROM(FUNCTION-RECEIVED)
*                    END-EXEC.
     CALL 'MQCONN'  USING
                          CONN-QM-NAME-AREA
                          CONN-HCONN-ADDR-AREA
                          CONN-CCODE-ADDR-AREA
                          CONN-RCODE-ADDR-AREA.
      MOVE  PARM-RECEIVED  TO DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(51) FROM(FUNCTION-RECEIVED)
*                    END-EXEC.
     IF  CONN-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
               XXXX-ERROR.
  2000-CONNECT-EXIT.
```

```
        EXIT.
        EJECT
*
*-----------------------------------------------------------------*
 2100-OPEN.
*-----------------------------------------------------------------*
*    EXEC CICS ENTER TRACEID(52) FROM(FUNCTION-RECEIVED)
*                   END-EXEC.
     CALL 'MQOPEN'  USING
                          OPEN-HCONN-ADDR-AREA
                          OPEN-Q-NAME-AREA
                          OPEN-Q-OPEN-OPTIONS
                          OPEN-HOBJ-ADDR-AREA
                          OPEN-CCODE-ADDR-AREA
                          OPEN-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(53) FROM(FUNCTION-RECEIVED)
*                   END-EXEC.
     IF  OPEN-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
               XXXX-ERROR.
 2100-OPEN-EXIT.
        EXIT.
        EJECT
*
*-----------------------------------------------------------------*
 2200-PUT.
*-----------------------------------------------------------------*
*    EXEC CICS ENTER TRACEID(54) FROM(FUNCTION-RECEIVED)
*                   END-EXEC.
*  PUT TO QUEUE
     CALL 'MQPUT'   USING
                          PUT-HCONN-ADDR-AREA
                          PUT-HOBJ-ADDR-AREA
                          PUT-MSG-DESCRIPTOR
                          PUT-PUT-OPTIONS
                          PUT-BUFFER-L-AREA
                          PUT-BUFFER-AREA
                          PUT-CCODE-ADDR-AREA
                          PUT-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(55) FROM(FUNCTION-RECEIVED)
*                   END-EXEC.
     IF  PUT-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
               XXXX-ERROR.
 2200-PUT-EXIT.
        EXIT.
*-----------------------------------------------------------------*
  2300-GET.
*-----------------------------------------------------------------*
*  GET TO QUEUE TO QM
*    EXEC CICS ENTER TRACEID(56) FROM(FUNCTION-RECEIVED)
*                   END-EXEC.
     CALL 'MQGET'   USING
                          GET-HCONN-ADDR-AREA
                          GET-HOBJ-ADDR-AREA
                          GET-MSG-DESCRIPTOR
                          GET-GET-OPTIONS
                          GET-BUFFER-L-AREA
                          GET-BUFFER-AREA
```

```
                        GET-DATA-L-AREA
                        GET-CCODE-ADDR-AREA
                        GET-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
     EXEC CICS SYNCPOINT    END-EXEC.
*    EXEC CICS ENTER TRACEID(57) FROM(FUNCTION-RECEIVED)
*                END-EXEC.
     IF  GET-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
              XXXX-ERROR.
  2300-GET-EXIT.
        EXIT.
*---------------------------------------------------------------*
  2400-PUT1.
*---------------------------------------------------------------*
*  PUT1 QUEUE TO QM.
*    EXEC CICS ENTER TRACEID(58) FROM(FUNCTION-RECEIVED)
*                END-EXEC.
     CALL 'MQPUT1'  USING
                        PUT1-HCONN-ADDR-AREA
                        PUT1-Q-NQME-AREA
                        PUT1-MSG-DESCRIPTOR
                        PUT1-PUT-OPTIONS
                        PUT1-BUFFER-L-AREA
                        PUT1-BUFFER-AREA
                        PUT1-CCODE-ADDR-AREA
                        PUT1-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(59) FROM(FUNCTION-RECEIVED)
*                END-EXEC.
     IF  PUT1-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
              XXXX-ERROR.
     EXEC  CICS SYNCPOINT
        END-EXEC.
*
  2400-PUT1-EXIT.
        EXIT.
*
*---------------------------------------------------------------*
  2500-INQUIRE.
*---------------------------------------------------------------*
*    EXEC CICS ENTER TRACEID(60) FROM(FUNCTION-RECEIVED)
*                END-EXEC.
     CALL 'MQINQ'   USING
                        INQ-HCONN-ADDR-AREA
                        INQ-HOBJ-ADDR-AREA
                        INQ-MQI-SECTOR-COUNT
                        INQ-MQI-SECTOR
                        INQ-MQI-IN-ATTR-COUNT
                        INQ-MQI-IN-ATTR
                        INQ-MQI-CHAR-ATTR-LENGTH
                        INQ-MQI-CHAR-ATTR
                        INQ-CCODE-ADDR-AREA
                        INQ-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(61) FROM(FUNCTION-RECEIVED)
*                END-EXEC.
     IF  INQ-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
              XXXX-ERROR.
  2500-INQUIRE-EXIT.
```

```
       EXIT.
*-------------------------------------------------------------*
  2600-CLOSE.
*-------------------------------------------------------------*
*  CLOSE  QUEUE
*    EXEC CICS ENTER TRACEID(62) FROM(FUNCTION-RECEIVED)
*                  END-EXEC.
     CALL 'MQCLOSE' USING
                          CLOSE-HCONN-ADDR-AREA
                          CLOSE-HOBJ-ADDR-AREA
                          CLOSE-Q-OPEN-OPTIONS
                          CLOSE-CCODE-ADDR-AREA
                          CLOSE-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(63) FROM(FUNCTION-RECEIVED)
*                  END-EXEC.
     IF  CLOSE-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
               XXXX-ERROR.
  2600-CLOSE-EXIT.
       EXIT.
*-------------------------------------------------------------*
  2700-DISCONNECT.
*-------------------------------------------------------------*
*
*  DISC  QUEUE
*    EXEC CICS ENTER TRACEID(64) FROM(FUNCTION-RECEIVED)
*                  END-EXEC.
     CALL 'MQDISC'  USING
                          DISC-HCONN-ADDR-AREA
                          DISC-CCODE-ADDR-AREA
                          DISC-RCODE-ADDR-AREA.
      MOVE PARM-RECEIVED  TO  DFHCOMMAREA.
*    EXEC CICS ENTER TRACEID(65) FROM(FUNCTION-RECEIVED)
*                  END-EXEC.
     IF  DISC-CCODE-VALUE NOT EQUAL ZERO   THEN GO TO
               XXXX-ERROR.
  2700-DISCONNECT-EXIT.
        EXIT.
*
*-------------------------------------------------------------*
 9900-NO-COMM-AREA.
*-------------------------------------------------------------*
*  EIBCALEN = 0    NO COMMON AREA RECEIVED.
     MOVE   'NOCOMMAR' TO FUNCTION-RECEIVED.
*     EXEC CICS ENTER TRACEID(91) END-EXEC.
        GO        TO   0000-RETURN.
 9900-NO-COMM-AREA-EXIT.
        EXIT.
```

Note: When developing the program, we found it difficult to debug in a CICS environment. The eventual solution was to insert TRACE requests, so that CEDF displays would be driven and each request could be readily identified, since each request has a different TRACEID. On entering CEDF we used the DISPLAY WORKING STORAGE function to check on the contents of working storage, including both XPCC buffers and also the ezBRIDGE Transact on VSE/ESA for MQSeries parameters. These trace requests have been left in the program, although they have been converted to comments.

E.3.2 EZBMTASK

```
         PRINT ON,GEN
* ************************************************************
*
*   PROGRAM-ID.  EZBMTASK
*
*   USAGE.       IS A LONG-RUNNING CICS TASK THAT ATTACHES A
*                SUBTASK TO DO XPCC CALLS TO THE BATCH SUBROUTINE.
*                THIS PROGRAM ALSO LINKS TO MQICICS THAT CALLS
*                THE MQ SERVICES REQUIRED.
*
* ************************************************************
EZBMTASK DFHEIENT CODEREG=(3,6)
         DFHREGS
         SPACE 3
         MVI   PECB,C' '               CLEAR PSEUDO-ECB
         SPACE 3
*
*  ATTACH SUBTASK TO DO XPCC ETC
         ATTACH EZBSTASK,SAVE=SAVEXPCC
         SPACE 3
*        PRINT ON,NOGEN
DELAYL   EQU   *
         MVC   TIMEOUT,=F'10'          SET LOOP TIMER
DELAY    EQU   *
         EXEC CICS DELAY FOR SECONDS(TIMEOUT)
         CLI   PECB,C'Q'               CHECK  PSEUDO-ECB FOR QUIT
         BE    ENDIT                   * IF SET - THEN EXIT
         CLI   PECB,C'S'               CHECK  PSEUDO-ECB
         BNE   DELAY                   * IF NOT SET - THEN WAIT
         SPACE 3
*                                      WORK TO DO SO ....
         MVC   TIMEOUT,=F'1'           * SET DELAY TIME LOW AS POSSIBLE
*
         EXEC CICS WRITE OPERATOR TEXT('ABOUT TO LINK')
*
         EXEC CICS LINK PROGRAM('MQICICS') COMMAREA(BUFF1)            X
               LENGTH(FOURK)
         MVI   PECB,C' '
         POST  SUBTECB
         CLC   0(8,R5),=CL8'EZDISC'
         BE    DELAYL
         CLC   0(8,R5),=CL8'MQDISC'
         BE    DELAYL
         B     DELAY
         SPACE 3
ENDIT    DS    0H
*
         EXEC CICS WRITE OPERATOR TEXT('END OF TASK')
*
         EXEC CICS ISSUE RESET
         EXEC CICS RETURN
PECB     DS    C
SAVEXPCC DS    15D
TIMEOUT  DS    F
BUFFERLN DS    F
SUBTECB  DC    F'00'
TIMERECB DC    F'00'
XPCCECB1 DC    F'00'
```

```
FOURK    DC    H'4096'
EIGHT    DC    H'08'
FOUR     DC    H'04'
FOURF    DC    F'04'
ZEROF    DC    F'00'
*
*  SUBTASK STARTS BELOW
*
EZBSTASK DS    0F
         WTO   'IN SUBTASK'
CONAGAIN DS    0H
         LA    R4,XPCCB1                POINT TO XPCCB
         USING IJBXPCCB,R4
         XPCC  XPCCB=(R4),FUNC=IDENT,   IDENTIFY                        X
               FDSCR=UNIQUE
         CH    R15,EIGHT                AM I UNIQUE?
         BNL   ERROR                    * NO .. ERROR
*        PRINT ON,GEN
         WTO   'IDENTIFIED'
         XPCC  XPCCB=(R4),FUNC=CONNECT,                                 X
               MECB=XPCCECB1
         LTR   R15,R15                  CONNECTED BOTH SIDES?
         BZ    CONOK                    *  YES - CONNECTION OK
         CH    R15,FOUR                 HALF CONNECTED?
         BNE   ERROR                    * NO .. ERROR
         WAIT  XPCCECB1
*        PRINT ON,GEN
         WTO   'CONNECTED'
CONOK    DS    0H
         WTO   'ABOUT TO WAIT FOR RECEIVE'
         LA    R2,IJBXRECB
         WAIT  (R2)                     WAIT FOR A MESSAGE
         MVC   IJBXRECB,ZEROF
         XPCC  XPCCB=(R4),              RECEIVE REQUEST                 X
               FUNC=RECEIVE
         LTR   R15,R15
         BNZ   RECERR
         SPACE 3
         WTO   'RECEIVED'
         LA    R5,BUFF1                 ADDRESS BUFFER
         CLC   0(8,R5),=CL8'EZQUIT'     MUST WE QUIT?
         BE    DISCONCT
         CLC   0(8,R5),=CL8'EZDISC'     END OF RUN?
         BE    DISCONCT
         MVI   PECB,C'S'                SET PSEUDO-ECB
         WTO   'WAITING FOR REDISPATCH'
         WAIT  SUBTECB                  * AND WAIT TO BE REDISPATCHED
         MVC   SUBTECB,ZEROF            * RESET ECB
         LA    R4,XPCCB1                POINT TO XPCCB
         XPCC  XPCCB=(R4),              SEND DATA BACK                  X
               FUNC=SEND
         LA    R2,IJBXSECB
         WAIT  (R2)                     SEND COMPLETE
         MVC   IJBXSECB,ZEROF
         LTR   R15,R15
         BNZ   SENDERR
         SPACE 3
         WTO   'SENT'
```

```
         B     CONOK
DISCONCT DS    OH
         XPCC  XPCCB=(R4),              TERMINATE CONNECTION          X
               FUNC=DISCONN
         WTO   'DISCONNECTED'
         XPCC  XPCCB=(R4),              STOP XPCC                     X
               FUNC=TERMIN
         WTO   'TERMINATED'
         MVC   XPCCECB1,ZEROF
         MVC   IJBXRECB,ZEROF
         MVC   IJBXSECB,ZEROF
         LA    R5,BUFF1                 ADDRESS BUFFER
*        LA    R5,100(0,R5)             POINT AT PARAMETER AREA
         CLC   0(8,R5),=CL8'EZQUIT'     MUST WE QUIT?
         BE    CANCEL
         SETIME 3,TIMERECB              WAIT TO GIVE OTHER TASKS TIME
         WAIT  TIMERECB                 * TO GET OUT
         B     CONAGAIN                 *  AND RE-CONNECT
*
ERROR    DS    OH
         LA    R15,21
CANCEL   DS    OH
         MVI   PECB,C'Q'                TELL CICS TASK TO QUIT
         DETACH                         STOP SUBTASK
*        PRINT ON,GEN
XPCCB1   XPCCB APPL=EZBAPPLC,TOAPPL=ANY,                             X
               BUFFER=(BUFF1,L'BUFF1),VERSION=2
*        PRINT ON,NOGEN
*
SENDERR  DS    OH
         WTO   'ERROR IN SEND'
         B     ERROR
RECERR   DS    OH
         WTO   'ERROR IN RECEIVE'
         B     ERROR
*
         DS    OF
BUFF1    DS    CL4096
         MAPXPCCB VERSION=2
*
         END
```

E.4 Jobs and Programs to Shut Down the System

The components that are needed for an orderly shutdown are:

1. A CICS program, specified in the PLTSD which uses SVC30 to start a batch job.
2. Batch JCL to run the job which will request termination of the online side of the interface, and will also clear up the batch side.
3. The source code for the batch program TASKCAN.

E.4.1 STOPXPCC

```
* ****************************************************************
*
*   PROGRAM-ID.  STOPXPCC
*
*   USAGE.       IS RUN AS PART OF THE CICS SHUTDOWN, BY BEING
*                DEFINED IN THE CICS PLTSD. RELEASES A JOB CALLED
*                TASKCAN THAT SHUTS DOWN XPCC IN THE CICS REGION
*                AND TERMINATES THE TRANSACTION.
*
* ****************************************************************
         TITLE 'SVC30  CICS/VSE PROGRAM TO SHUT DOWN XPCC SUBTASK'
STOPXPCC CSECT
         BALR  10,0                    INITIALIZE BASE REGISTER
         USING *,10                    ESTABLISH ADDRESSABILITY
         XR     0,0                    SETUP INPUT REGISTER
         LA     1,PARMLST              AND POINT TO PARAMETER LIST
         SVC    30                     ISSUE THE COMMAND
         EXEC   CICS RETURN
PARMLST  DC    XL2'000E',AL4(INBUF)    PARMLIST FOR SVC30
INBUF    DC    CL14'R RDR,TASKCAN'
         END
```

E.4.2 Job Control to Run the Batch Program to Shut Down the Interface

```
* $$ JOB JNM=TASKCAN,DISP=L,CLASS=A
* $$ LST DISP=D,CLASS=A,PRI=5
// JOB TASKCAN
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(MQMUSR1.RICH,PRD2.DBASE,PRD2.CICSR)
// EXEC TASKCAN,SIZE=512K
/*
/&
* $$ EOJ
```

E.4.3 Program to Allow Orderly Termination of the Interface

```
         PRINT ON,GEN
* ****************************************************************
*
*   PROGRAM-ID.  TASKCAN
*
*   USAGE.       IS RELEASED BY THE STOPXPCC PROGRAM WHICH IS
*                DEFINED RUN IN THE CICS PLTSD TO SHUT DOWN THE
*                XPCC CONNECTION ON THE CICS SIDE AND TERMINATE
*                THE CICS TRANSACTION.
*
* ****************************************************************
TASKCAN  CSECT
         DFHREGS
* ****************************************************************
*   ESTABLISH ADDRESSABILITY
* ****************************************************************
         STM   14,12,12(13)            SAVE CALLERS REGISTERS
         USING *-4,3,6                 USE R3 AND R6 AS BASE REGS
         LR    3,15                    LOAD BASE REG
         LR    6,3                     LOAD SECONDARY BASE REG
         AH    R6,FOURK                ADD 4096
         LR    R10,R1                  SAVE R1 FOR LATER USE
```

```
        SPACE 3
* ********************************************************************
*  XPCC INITIALISATION STARTS
* ********************************************************************
        LA    R4,XPCCB1               POINT TO XPCCB
        USING IJBXPCCB,R4
        XPCC  XPCCB=(R4),FUNC=IDENT,  IDENTIFY                      X
              FDSCR=UNIQUE
        CH    R15,EIGHT               AM I UNIQUE?
        BNL   ERROR                   * NO .. ERROR
        SPACE 3
        WTO   'IDENTIFIED'
        SPACE 3
*       PRINT ON,GEN
        XPCC  XPCCB=(R4),FUNC=CONNECT,                              X
              MECB=XPCCECB1
        LTR   R15,R15                 CONNECTED BOTH SIDES?
        BNZ   ERROR                   * NO .. ERROR IN CONNECTION
        SPACE 3
        WTO   'CONNECTED'
        SPACE 3
*
        LA    R5,BUFF1                ADDRESS BUFFER
        LA    R5,100(0,R5)            POINT AT PARAMETER AREA
        MVC   0(8,R5),=CL8'EZQUIT'    TELL TASK YOU WANT TO QUIT
* ********************************************************************
*  XPCC SEND
* ********************************************************************
        WTO   'IN XPCCWORK'
        XPCC  XPCCB=(R4),             SEND DATA BACK                X
              FUNC=SEND
        LA    R2,IJBXSECB
        WAIT  (R2)                    SEND COMPLETE
        WTO   'SENT'
*  XPCC DISCONNECT ETC
* ********************************************************************
        XPCC  XPCCB=(R4),             TERMINATE CONNECTION          X
              FUNC=DISCONN
        WTO   'DISCONNECTED'
        XPCC  XPCCB=(R4),             STOP XPCC                     X
              FUNC=TERMIN
        WTO   'TERMINATED'
*
        SR    R15,R15
ERROR   DS    0H
*       LM    14,12,12(13)           RESTORE CALLERS REGS
        EOJ   RC=(15)                RETURN
*
*
*
*       PRINT ON,GEN
XPCCB1  XPCCB APPL=EZBAPPLB,TOAPPL=EZBAPPLC,                        X
              BUFFER=(BUFF1,L'BUFF1),VERSION=2
XPCCECB1 TECB
FOURK   DC    H'4096'
EIGHT   DC    H'08'
*
        DS    0F
BUFF1   DS    CL4096
```

```
         MAPXPCCB VERSION=2
*
         END
```

Glossary

browse.. In message queuing, to copy a message without removing it from the queue. See also get.

dead-letter queue.. A queue to which a queue manager or application sends messages that it cannot deliver to their correct destination.

distributed queue management.. In message queuing, the setup and control of message channels to queue managers on other systems.

get.. In message queuing, to retrieve a message by removing the message from a queue or by browsing the message. See also browse.

initiation queue.. A local queue on which the queue manager puts trigger messages.

local definition.. An MQM object that belongs to a local queue manager.

local definition of a remote queue.. An MQM object that belongs to a local queue manager. This object defines the attributes of a remote queue.

local queue.. A queue that belongs to the local queue manager. A local queue can contain a list of messages waiting to be processed. Contrast with remote queue.

MCA.. Message channel agent.

message.. (1) In message queuing applications, a communication sent from one program to another. (2) In system programming, information intended for the terminal operator.

message channel agent (MCA).. A program that transmits prepared messages from a transmission queue to a communication link, or from a communication link to a destination queue.

message queue.. Synonym for queue.

Message Queue Interface (MQI).. The programming interface provided by the MQSeries message queue managers. This programming interface allows application programs to access message queuing services.

message queuing.. A programming technique in which each program within an application communicates with the other programs by putting messages on queues.

messaging.. A method for communication between programs. Messaging can be synchronous or independent of time.

MQI.. Message Queue Interface.

MQM.. Message Queue Manager.

object.. In MQM, objects define the attributes of queue managers, queues, process definitions, and namelists.

persistent message.. A message that survives a restart of the queue manager.

queue.. An MQM object. Message queuing applications can put messages on, and get messages from, a queue. A queue is owned and maintained by a queue manager. Queues can be of type local, alias, model, or remote. Local queues can contain a list of messages waiting to be processed. Queues of other types cannot contain messages--they point to other queues.

queue manager.. (1) A system program that provides queuing services to applications. It provides an application programming interface so that programs can access messages on the queues that the queue manager owns. (2) An MQM object that defines the attributes of a particular queue manager.

queuing.. See message queuing.

remote queue.. A queue that belongs to a remote queue manager. Programs can put messages on remote queues, but they cannot get messages from remote queues. Contrast with local queue.

rollback.. Synonym for back out.

transmission queue.. A local queue on which prepared messages destined for a remote queue manager are temporarily stored.

trigger event.. An event (such as a message arriving on a queue) that causes a queue manager to create a trigger message on an initiation queue.

trigger message.. A message that contains information about the program that a trigger monitor is to start.

trigger monitor.. A continuously-running application that serves one or more initiation queues. When a trigger message arrives on an initiation queue, the trigger monitor retrieves the message. It uses the information in the trigger message to start a process that serves the queue on which a trigger event occurred.

triggering.. In MQM, a facility that allows a queue manager to start an application automatically when predetermined conditions on a queue are satisfied.

undelivered message queue.. See dead-letter queue.

unit of recovery.. A recoverable sequence of operations within a single resource manager (such as MQM MVS/ESA). Compare with unit of work.

unit of work.. A recoverable sequence of operations performed by an application between two points of consistency. A unit of work begins when a transaction starts or at a user-requested syncpoint. It ends either at a user-requested syncpoint or at the end of a transaction. Compare with unit of recovery.

List of Abbreviations

API	Application Programming Interface
APPC	Advanced Program to Program Communication
ASCII	American Standard Code for Information Interchange
CI	Control Interval
CICS	Customer Information Control System
CSD	CICS System Definition
EBCDIC	Extended Binary Coded Decimal Interchange Code
ECB	Event Contol Block
IBM	International Business Machines Corporation
ISC	Inter System Connection
ITSO	International Technical Support Organization
IVP	Installation Verification Procedure
JCL	Job Control Language
JECL	Job Entry Control Language
LU	Logical Unit
MCA	Message Channel Agent
MQ	Messaging and Queuing
MQI	Messaging and Queuing Interface
MQM	Messaging and Queuing Manager
PCT	Program Control Table
PLTPI	Program List Table for Program Initialization
PLTSD	Program List Table for Shut Down
PPT	Processing Program Table
SVC	Supervisor Call
TCB	Task Control Block
VSAM	Virtual Storage Access Method
VSE	Virtual System Enhanced
VTAM	Virtual telecommunications Access Method
WTO	Write To Operator
XCTL	Transfer Control (a CICS command)
XECB	Cross Partition Event Control Block
XPCC	Cross Partition Communication Control

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

ITSO Technical Bulletin Evaluation RED000

Messaging and Queuing Extensions for VSE/ESA

Publication No. GG24-4296-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

_____ _____
Name Address

_____ _____
Company or Organization

_____ _____
Phone No.

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM International Technical Support Organization
Department 3222, Building 71032-02
POSTFACH 1380
71032 BOEBLINGEN
GERMANY

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.