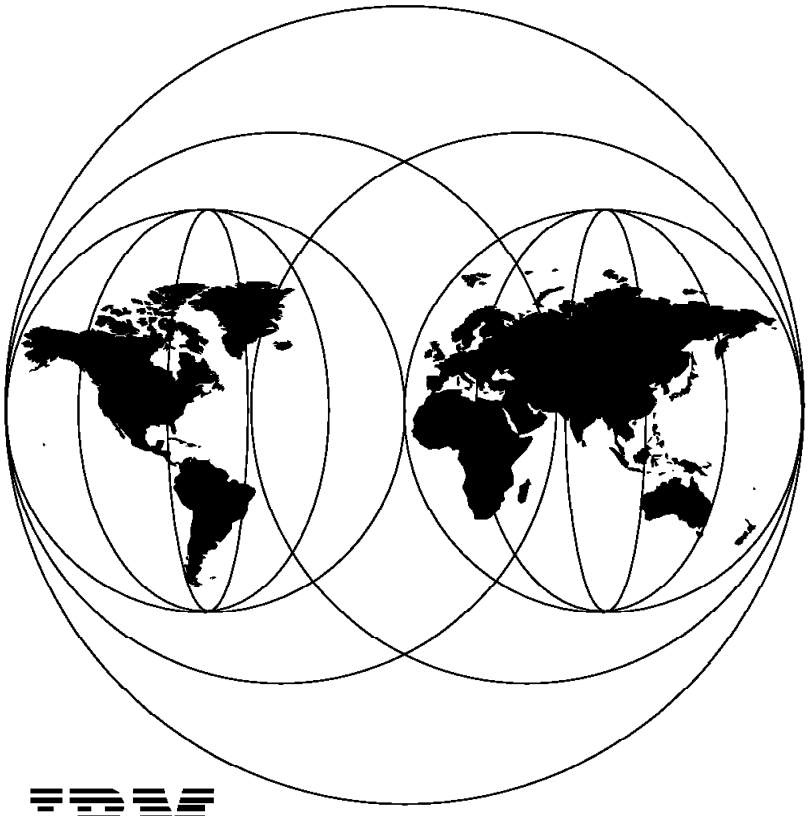
SG24-4882-00

Examples of Using MQSeries on WWW

November 1996



IBM
International Technical Support Organization
Raleigh Center

International Technical Support Organization

SG24-4882-00

Examples of Using MQSeries on WWW

November 1996

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special Notices” on page 199.

First Edition (November 1996)

This edition applies to the MQSeries family of products. Although MQSeries includes a number of products, this project involved use of MQM AIX, MQM OS/2, MQM NT, and MQM MVS/ESA.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Figures

Preface

This redbook describes an approach for using the Internet common gateway interface (CGI) through use of an MQSeries application. All steps are documented that must be done to enable a complete scenario including creating the GUI front-end by using HTML documents and writing a Web-enabled MQSeries application which communicates with MQSeries application servers on different platforms. The MQSeries Internet Gateway SupportPac was used as a base for work done in creating this redbook.

This redbook was written for persons who have need of using services of the internet while also involving MQSeries applications to meet a systemwide application need.

Some knowledge of application programming and development is assumed.

How This Redbook Is Organized

This redbook contains 207 pages. It is organized as follows:

- Chapter 1, “Background”

  This chapter describes an internet gateway and summarizes the common gateway interface.

- Chapter 2, “MQSeries Internet Gateway SupportPac”

  This chapter discusses in further detail the MQSeries Internet Gateway SupportPac including how the MQSeries Internet Gateway SupportPac is organized and how it can be used.

- Chapter 3, “Example of MQSeries Web Usage using User Written CGI”

  This chapter shows examples of using the MQSeries Internet Gateway SupportPac from a CGI-based application.

- Chapter 4, “Example of Using the MQSeries Internet Gateway SupportPac with WTPING/WTPONG”

  This chapter shows an ITSO WTPING/WTPONG sample driven by a Web browser using the MQSeries Internet Gateway SupportPac and, also, a sample Web file-browser application using the MQSeries Internet Gateway SupportPac.

- The Appendix includes:

  Appendix A, “Source Listing and Compile Procedures”

  Appendix B, “MQSeries Definitions”

  Appendix C, “Source Listing and Compile Procedures”

  Appendix D, “Source Listing and Compile Procedures”

  Appendix E, “MQSeries Definitions”

  Appendix F, “Downloading the Internet Connection Server for OS/2 and MQSeries Internet Gateway Support Pac”

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization Raleigh Center.

The authors of this redbook were:

Joerg Bartel	**Young Pyo**
IBM Germany	**IBM Korea**

Contributing to this work included:

Pervez Goiporia	Frans Meij
IBM India	IBM Netherlands

The advisors of this project were:

Dave Shogren Dieter Wackerow
both of the International Technical Support Organization, Raleigh Center

Thanks to Trevor Cross and John Kelly of IBM Hursley for their invaluable contribution to this project.

Some base material in this document was obtained from various IBM forums and conference disks, including MKTOOLS plus the IBM Hursley SupportPacs.

Request for Feedback

Readers of this document are encouraged to feed back any information or comments regarding *any* of the material in this document. Please send your comments to:

Dave Shogren or Dieter Wackerow
ITSO-Raleigh
VNET: SHOGREN at WTSCPOK or WACKEROW at WTSCPOK

or: IBM Corporation HZ8D/B678/D100
Attn: Dave Shogren / Dieter Wackerow
Building 678 Rm D100
1001 Winstead Dr.
Raleigh (Cary) NC 27513

INTERNET: shogren@vnet.ibm.com
wackerow@vnet.ibm.com

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Background

This chapter gives an overview of the MQSeries Internet Gateway SupportPac and the common gateway interface.

1.1 Why an Internet Gateway for MQSeries

Because more and more customers are looking forward to using Internet technologies for connecting their users via Internet or Intranet, a communication bridge between the fancy but low-cost browser front-ends and operational backend servers is needed. An end user using a WWW browser workstation as in Figure 1 on page 4 need not know MQSeries is being used by applications which are running on the Internet connection server.

This is why IBM Hursley started to develop an MQSeries Internet Gateway SupportPac which uses the Web server CGI interface in order to connect a Web browser front-end with a Web server application which then uses MQSeries APIs to communicate with MQ-enabled applications on other platforms.

This gateway is made available as a SupportPac for OS/2 and AIX. It also includes application samples to demonstrate the functionality.

The possibility of coupling the Internet technology with the robust messaging technology of MQSeries benefits all WWW users who have to be interconnected with operational application servers within our customer installations.

Connecting MQSeries into Web server applications helps to isolate Web users from the internal network of the customers. It also helps to use Internet technology in the existing Intranets of our customers. A big benefit is the reuse of the same messaging interfaces to application servers from the Web as from internal applications.

1.1.1 SupportPac

SupportPacs are pieces of code or documentations and reports that are made available for the CICS and MQSeries products by the Hursley Lab.

They can be accessed through the Internet via the IBM Hursley home page http://www.hursley.ibm.com or internally by using the package TXPPACS on the MKTTOOLS disk.

1.2 Internet Common Gateway Interface

The Internet common gateway interface (CGI) is the way the Web server can communicate with other programs running on the server.

With CGI, the Web server can call up a program while passing parameters to that program. Examples of such passed information could be WWW browser-generated data supplied in HTML form passed onto the Internet connection server using HTTP protocol (refer to Figure 1 on page 4).

The called program then processes the data with its own business logic and the server passes the program's results back to the Web browser again using HTTP/HTML.

1.3 What This WWW Example Is Not

This document's WWW browser example of Web access to MQSeries application is only that: an example of using the Web browser, HTTP, Internet connection server, and MQSeries Internet Gateway SupportPac support. It is *not* an example of using MQSeries-to-MQSeries systems across a Web network.

Chapter 2. MQSeries Internet Gateway SupportPac

This chapter discusses in further detail the MQSeries Internet Gateway SupportPac.

2.1 Architecture

Web users connect to their Web servers and backend application servers via Internet or Intranet by running a Web browser such as the IBM WebExplorer or Netscape for the GUI part of the application.

The business logic of the customer applications are running in the Web server or backend server environments in a secured run-time environment.

The MQSeries Internet Gateway SupportPac provides a bridge between the synchronous World Wide Web and asynchronous MQSeries applications. Interaction with the gateway is via HTML fill out form POST requests. The form needs to identify target queue and queue manager names that the application servicing the requests will be using. This configuration work is done on the Web server that runs the MQSeries Internet Gateway SupportPac. The MQSeries system administrator would provide the queue manager and queue names. Refer to MQSeries documentation including *MQSeries for AIX System Management Guide*, SC33-1373 or MQSeries OS/2 documents and the *MQSeries Internet Gateway SupportPac User Guide* which is provided with the SupportPac.

The MQSeries application receiving the request will need to be able to generate HTML pages to return to the gateway and the SupportPac provides example code to do this. Figure 1 on page 4 shows how the Web access to MQSeries applications is done.

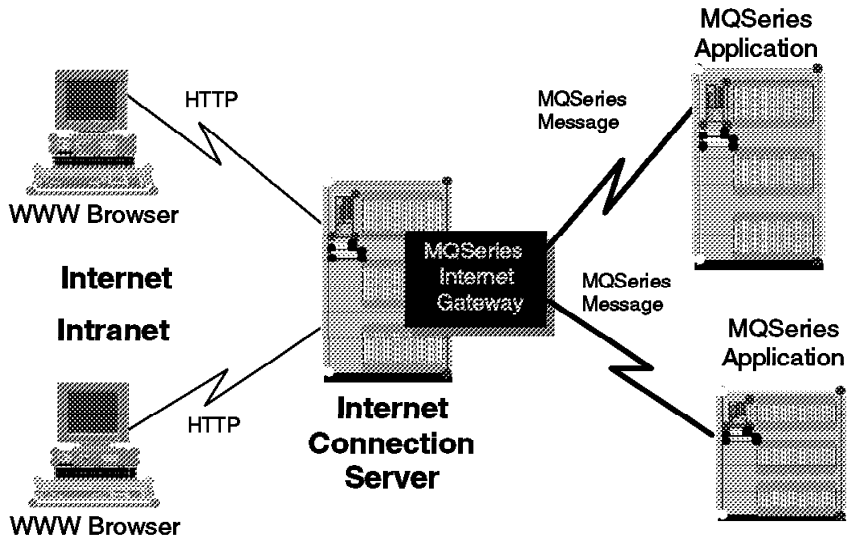


Figure 1. WWW Access

A big advantage of using MQSeries for accessing server applications is the reuse of existing interfaces to applications when connecting them into a new type of front-end such as the Web.

Building an application server infrastructure based on MQSeries allows for easy reuse of business logic functions.

In customer environments today, applications servers are deployed on different platforms. We have to reach application servers on multiple platforms for handling business requests of the end user. This can be seen as a logical transaction which may have many actual tasks. Figure 2 on page 5 summarizes this process.

In this scenario the MQSeries Internet Gateway SupportPac fits well. A Web-enabled MQSeries program running on the Web server provides connectivity to multiple backend applications. This is handled easily, as in any normal MQSeries application.

It is also possible to support multi-user access to the applications.

Another advantage is that MQSeries software need not be distributed to the end user workstation. This makes the total customer software distribution challenge easier, since application objects must only distributed to the server platforms. In addition, system management of the MQSeries environment becomes easier because only server platforms need to be directly managed.

The following figure shows how the MQSeries Internet Gateway SupportPac manages the coordination of multiple application servers.

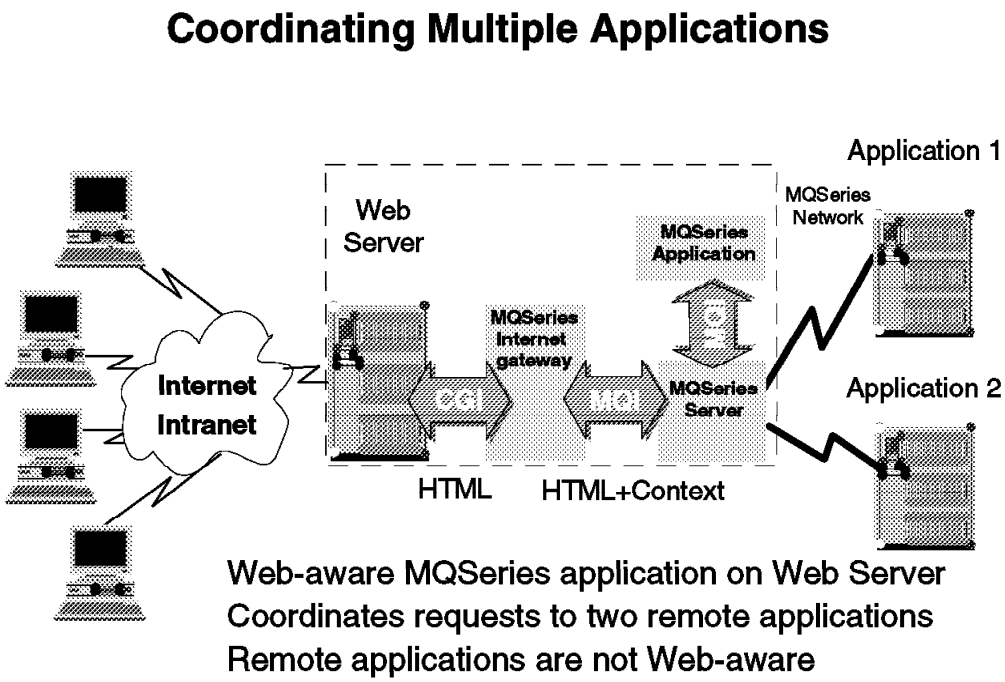


Figure 2. Coordinating Multiple Applications

2.2 Implementation of a Web-Enabled MQSeries Application

Two examples were used in this project. Our first example took the design of the MQSeries Internet Gateway SupportPac and some samples which are provided with the SupportPac as building blocks to write our own program designed to show how a Web user interacts with an MQSeries program running on the Web Server under OS/2 via an HTML form.

This Web-enabled program receives parameters via the CGI interface and sends parameter messages using MQSeries to two application servers. The parameters originate from the HTML form filled out by the end user on the WWW browser machine (refer to Figure 1 on page 4).

One application server has an AIX program running on a RISC System/6000 and the other application server has a CICS/ESA program running under MVS on a S/390. Both application server programs implement MQSeries.

The Web-enabled program running on the Web server handles the reply messages coming back from the remote application servers and sends back the output via a dynamically built HTML page. Presenting the data back to the Web browser is an application-dependant matter. In our example, the MQSeries calls were done in such a way that the data is not provided to the Web browser until all the application servers have been heard from or an error (timeout) occurs for one or more application servers.

The second example in this project used the MQSeries Internet Gateway SupportPac programs as they are provided by the SupportPac.

2.2.1 Software and Hardware Used in this Example

We installed our Web browser and Web server environment PCs with OS/2 Warp Connect with TCP/IP and Communication Manager/2 already installed.

For the application server platforms, we used a RISC System/6000 running AIX Version 3.2.5 with MQSeries Version 2 and a S/390 MVS machine running CICS/ESA 3.3 and MQSeries/ESA 1.1.4. C++ was used for application programs on AIX and OS/2. COBOL was used for the MVS application programs.

We used the IBM WebExplorer of the OS/2 Warp Connect package to retrieve the IBM Internet Connection Server for OS/2 and the MQSeries Internet Gateway SupportPac over the Internet.

Appendix F, “Downloading the Internet Connection Server for OS/2 and MQSeries Internet Gateway Support Pac” on page 189 shows how the IBM Internet Connection Server for OS/2 and the MQSeries SupportPac MA81 for OS/2 was obtained.

2.2.2 Basic Design of the Application

After successful installation of the Web server and the MQSeries Internet Gateway SupportPac we started to design our application.

We wanted to create a catalog-display application where images of cars and a corresponding descriptive text belonging to the car would be shown to the end user.

The Web-enabled MQSeries application program receives a selection key entered by the end user.

The Web-enabled program does an MQPUT and sends the key value in an MQSeries message to the AIX application server. The server program retrieves the correct image file name and sends it back in a reply MQSeries message.

The Web-enabled program also MQPUTs the key value to our CICS application server and the CICS program sends a corresponding descriptive text back as a reply.

After sending out those request messages, our Web-enabled MQSeries program retrieves the reply messages correlated to the requests which were sent to the application servers.

After analyzing the replies the Web-enabled program dynamically creates the reply HTML page and sends it back to the Web browser. The application is designed to be aware of late message arrival or missing replies; either a correlating reply value or an error message is displayed in the HTML page.

Here is the application configuration for the example.

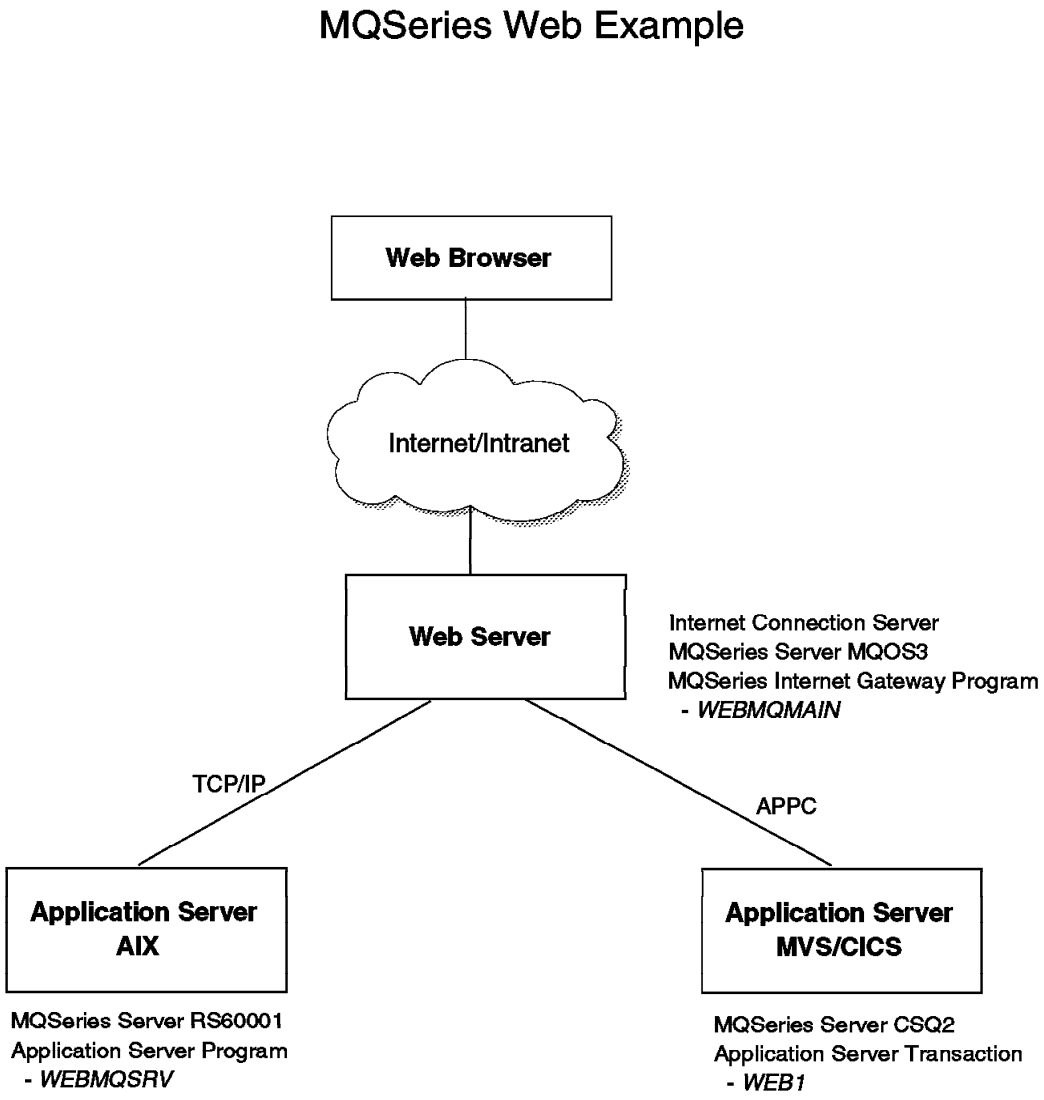


Figure 3. Web Application Configuration

Here is the MQSeries configuration for the example.

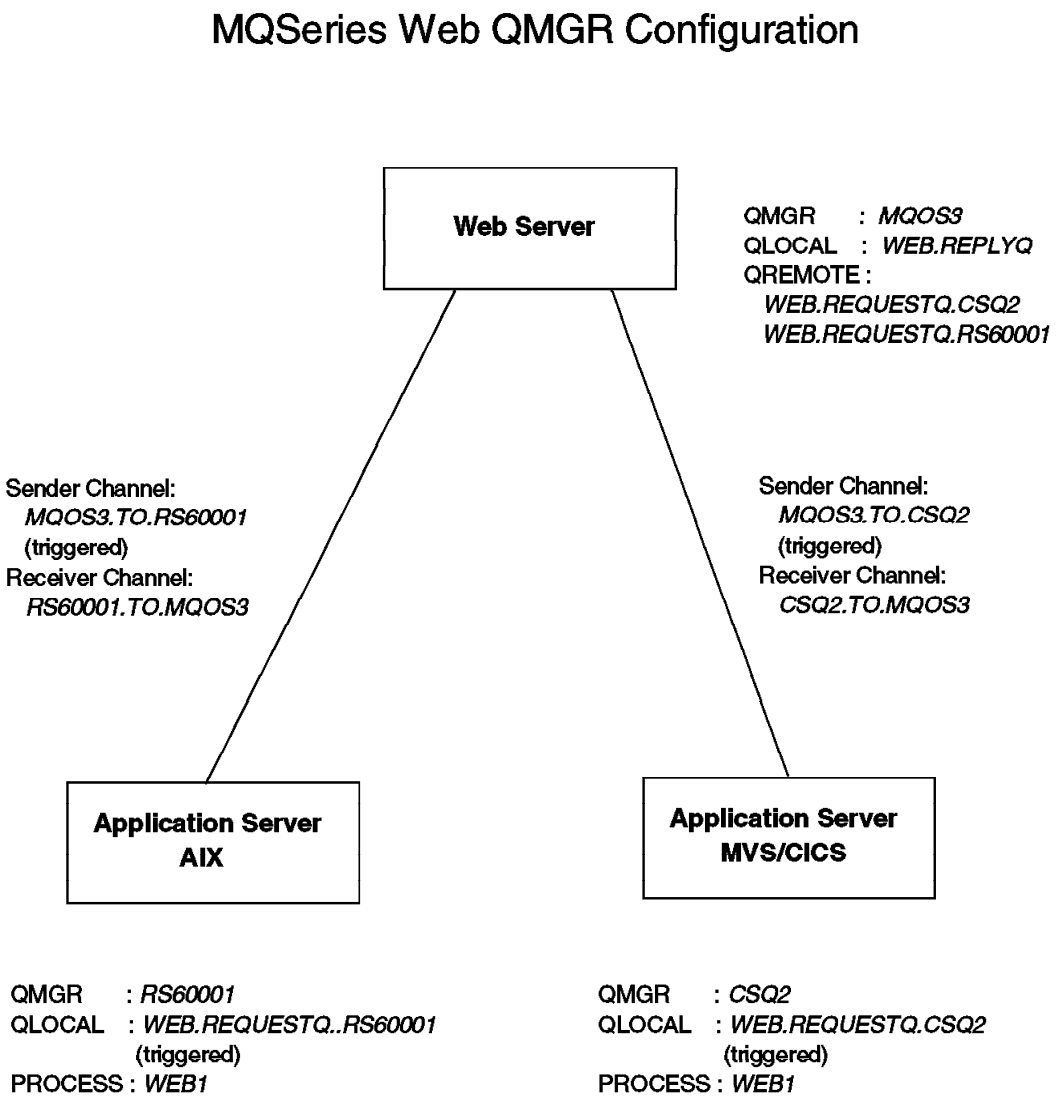


Figure 4. Web MQSeries Configuration

Here is the MQ-API and message flow for the example.

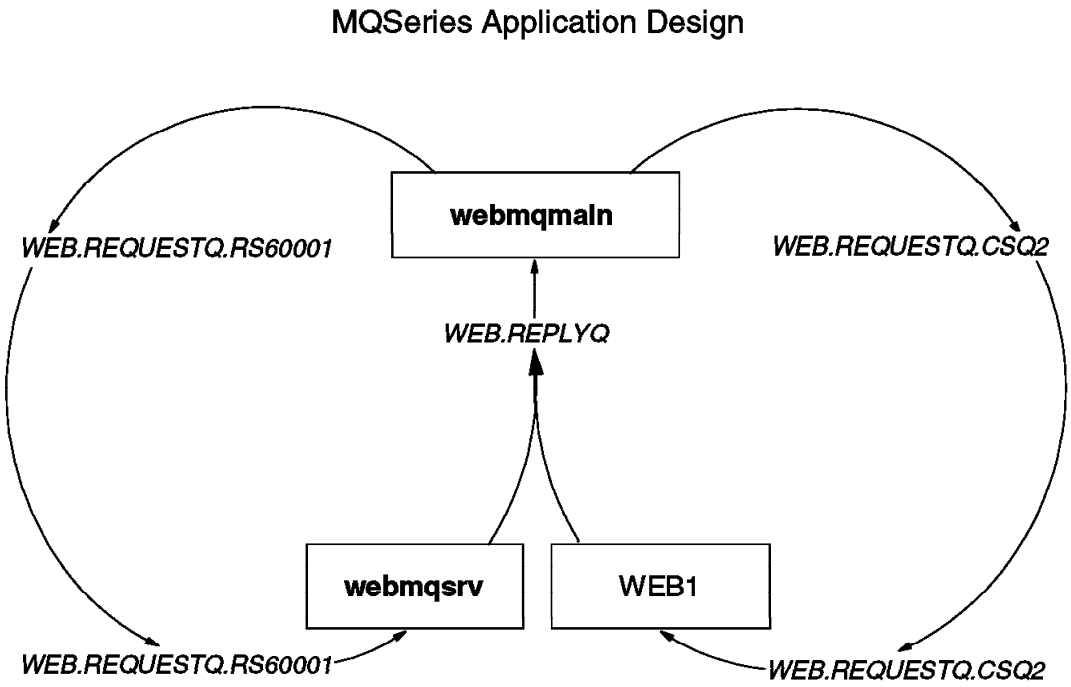


Figure 5. Web MQAPI Design

2.2.3 MQSeries Objects for the Application

Our application needs some MQSeries message queues to be defined to reach the backend application servers and get their replies back.

The local queue manager MQOS3 was already interconnected with the AIX queue manager RS60001 and the MVS queue manager CSQ2.

MQSeries channel definitions are found in Appendix B, “MQSeries Definitions” on page 87.

We defined a local reply queue WEB.REPLYQ on the OS/2 queue manager. This reply queue name was passed as ReplyToQ parameter on the request messages to the applications servers. For sending the requests messages to our application servers, we defined the following remote queues:

Request-Queue to CICS server: WEB.REQUESTQ.CSQ2
Request-Queue to AIX server: WEB.REQUESTQ.RS60001

On the target queue managers those queues were defined as local queues with triggering on. To enable application triggering on AIX we defined the local queue WEB1.INITQ and the process WEB1 which pointed to our application server program executable.

Then we started the trigger monitor with the command:

```
runmqtrm -q WEB1.INITQ.
```

The S/390 CICS trigger monitor was started as a normal process in our configuration.

We defined the process WEB1 for our application server to be triggered under CICS. Queue definitions and environment specific commands are found in Appendix B, “MQSeries Definitions” on page 87.

2.2.4 Application Programming

For the Web-enabled MQSeries program we used the following MQSeries Internet Gateway SupportPac sample programs from \MQGATE\SOURCE\SAMPLES as building blocks:

```
amqwput0.cpp
amqwget0.cpp
```

Our Web-enabled server application source code was named webmqmain.cpp and is shown in Figure 26 on page 57.

For the AIX application server, we used the following sample program from /usr/lpp/mqm/samp:

- `amqsecha.c`

Our AIX application server source code was named webmqsrv.c and is shown in Figure 31 on page 78.

For the S/390 CICS application server we used the following sample program:

- `MQM.V1R4.SCSQCOBS(CSQ4CVB1)`

Our CICS application server source code was named PRGQMIW1 with CICS tranid WEB1 and is shown in Figure 20 on page 32.

On OS/2 we used the C++ Compiler because the CGI functions require C++. The AIX program is written in C. The CICS application is written in COBOL II.

Compile the procedures, commands and source code found in Appendix A, “Source Listing and Compile Procedures” on page 31.

2.2.5 HTML Design

For the Web front-end we used the following MQSeries Internet Gateway SupportPac HTML samples as building blocks:

- amqwput.html in \MQGATE\HTDOCS

Our HTML objects are:

webmq.html
webmqh.html
MQGate.gif
credos.jpg
leo.jpg
avella2.jpg
pr-beta.jpg
newspo.jpg
pregio1.jpg
besta1.jpg
concept1.gif

Images for the cars and the descriptive texts were received via Internet from Web site:

http://wwww.kia.com/kia/product

2.2.6 MQSeries and WWW Security

In our scenario we used the WebExplorer and IBM Internet Connection Server for OS/2. In our own LAN environment we did not need the secured features of Web technology.

To run our application scenario in a secured fashion we could have installed the IBM WebExplorer with Secure Socket Layer (SSL) support or an equivalent Netscape browser together with the Internet Connection Secure Server for OS/2.

The secured versions of the Web products use SSL to create a secure channel between client and server and handle privacy, server authentication and integrity as summarized in Figure 6 on page 14.

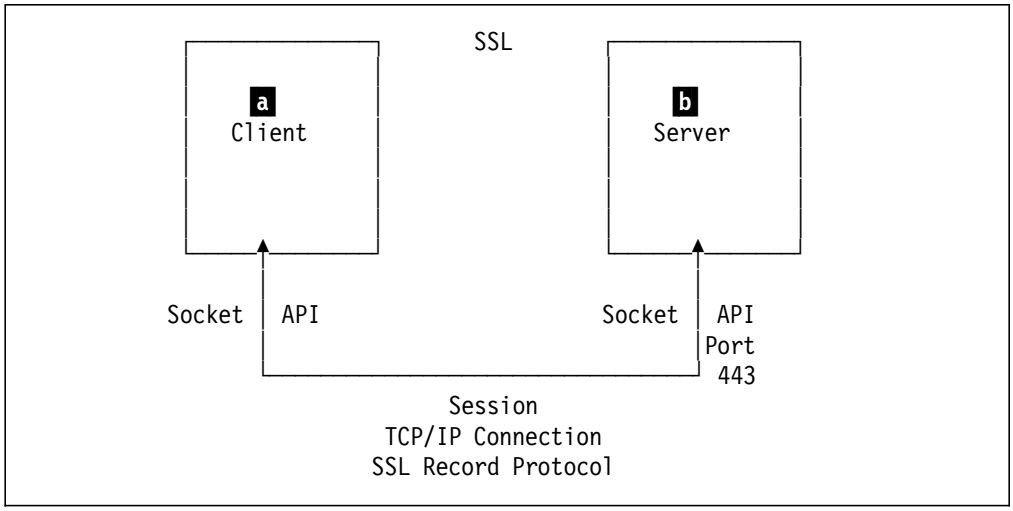


Figure 6. Secure Socket Layer (SSL) Overview

Those functions may be needed when accessing operational applications over the worldwide Internet using Web technology.

In our example, **a** is the WWW browser and **b** is the WWW server.

The data, once removed from the above indicated **a** and **b**, is in clear form; that is to say, in the examples used in this document the applications outside of **a** and **b** in Figure 6 on page 14 would be assumed secured in some other manner than SSL.

Chapter 3. Example of MQSeries Web Usage using User Written CGI

This chapter shows our first example of MQSeries on the Web server. In this example, we used the program samples provided by the MQSeries Internet Gateway SupportPac as a base for developing our own Web-enabled program.

All necessary code we built during our application design project is documented in the appendixes.

The following are the steps we took to run our application.

Before running the application we checked the conditions shown in Figure 7 on page 15. The Internet Connection Server was the server used in this project.

```
OS/2: Is the Internet Connection Server running?
      If not please start it!
      Is the queue manager MQOS3 running ?
      If not please start it!   (strmqm MQOS3)
      Is the channel initiator running ?
      If not please start it!   (runmqchi)
      Is the Listener running ?
      If not please start it!   (runmqlsr -t tcp)
      Is the COMMS-MGR running ?
      If not please start it!

AIX : Is the queue manager RS60001 running ?
      If not please start it!   (strmqm RS60001)
      Is the channel initiator running ?
      If not please start it!   (runmqchi)
      Is the Trigger monitor running ?
      If not please start it!   (runmqtrm -q WEB1.INITQ)

MVS : Is the queue manager CSQ2 running ?
      If not please start it!   (+start qmgr   )
      Is the channel initiator running ?
      We used CICS for DQM function, so check is CICS18 running
      If not please start it!   (s cics18)
      Is the Trigger monitor running ?
      It is automatically started with CICS.

Connectivity checklist:
      Do ping channel commands from OS/2 to AIX and vice versa.
      Do ping channel commands from OS/2 to MVS and vice versa.
```

Figure 7. Setup Checklist. All these steps are checked on the Web server and application servers.

After those basic checks, we run the application from the Web browser which uses the resources of the Web server (refer to Figure 1 on page 4).

On the Web browser:

Start IBM WebExplorer under OS/2.
Enter Web site http://mqos3/webmq.html.

Enter key ranging from 0000000001 to 0000000010.

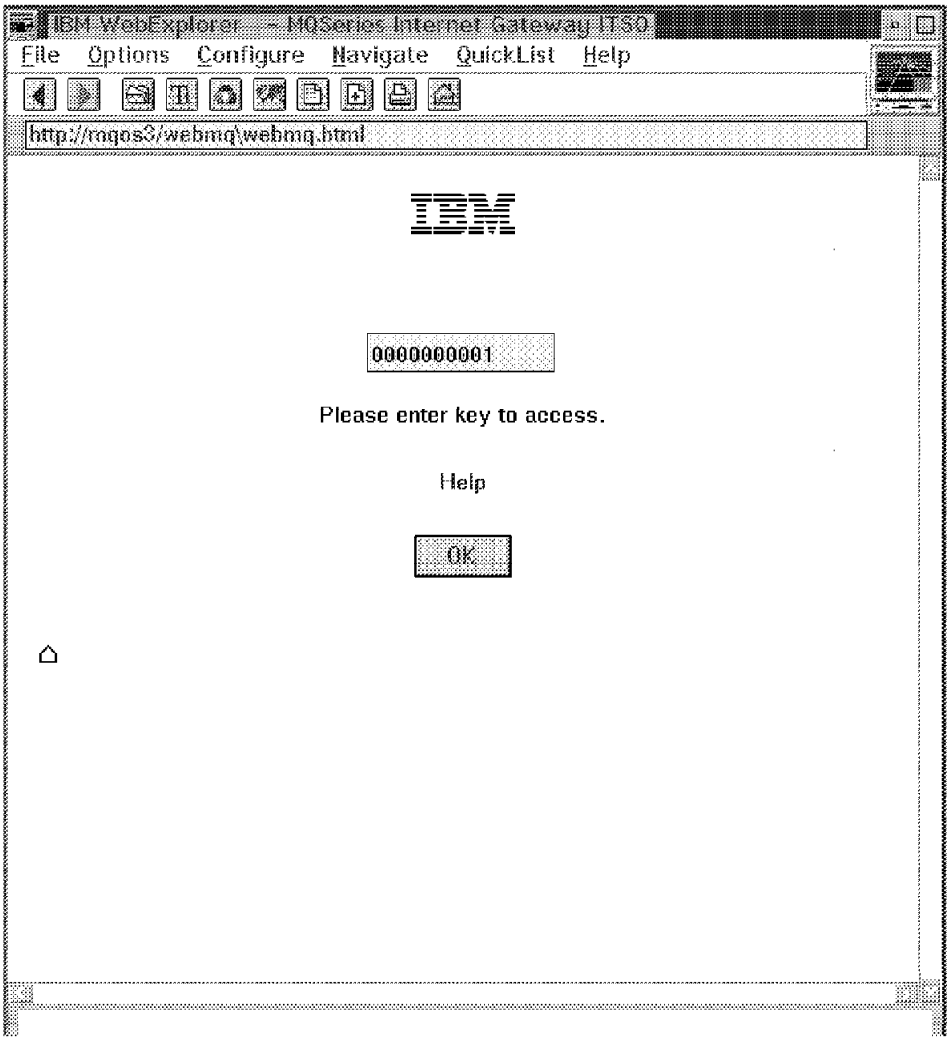


Figure 8. Web Application HTML

Click on **Help** as a sample for a help HTML page.

At this time we only wanted to demonstrate how to use a separate HTML page for help purposes. There is no real application help information on this page during our example. A.10, “HTMLPAGE2” on page 77 is the Help HTML page this project used. See Internet HTML documents for building HTML pages.

After displaying the Help page, go back to the previous page.

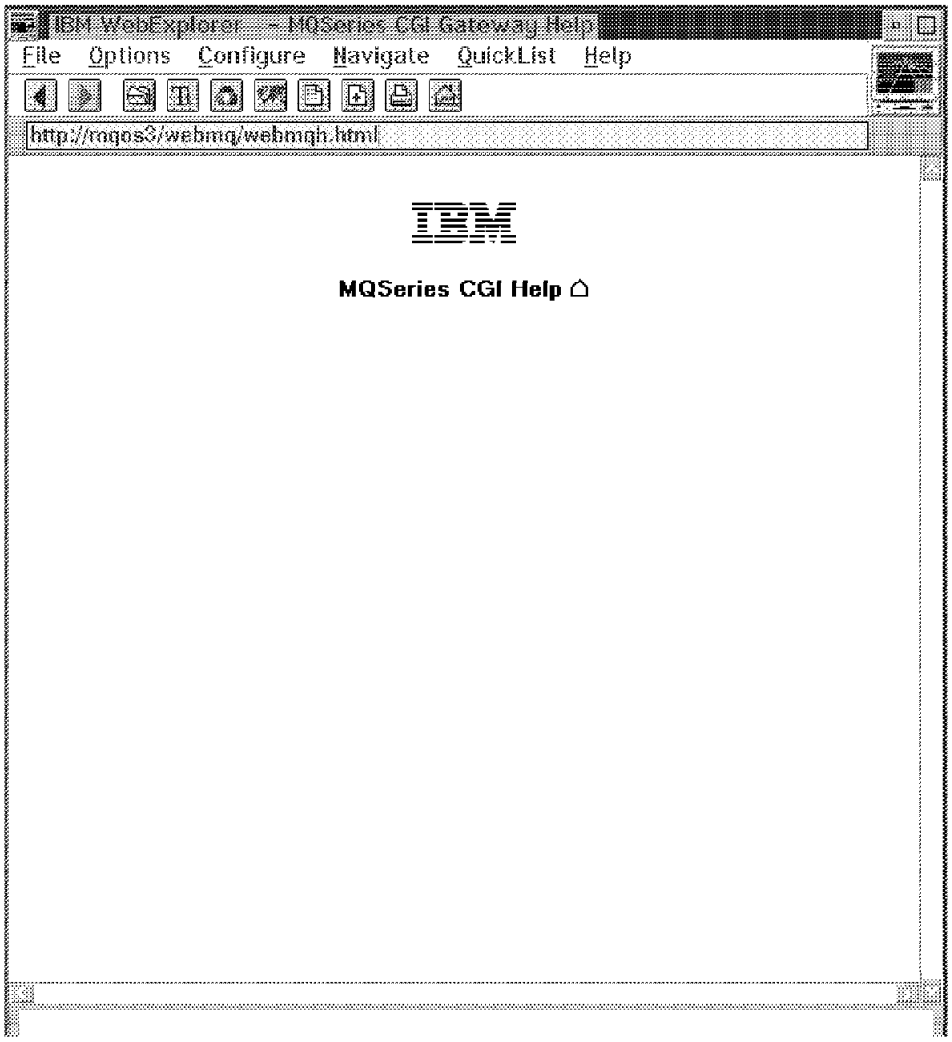


Figure 9. Web Application Help

Select the **OK** button after entering a key value.

The application runs and presents the following page.

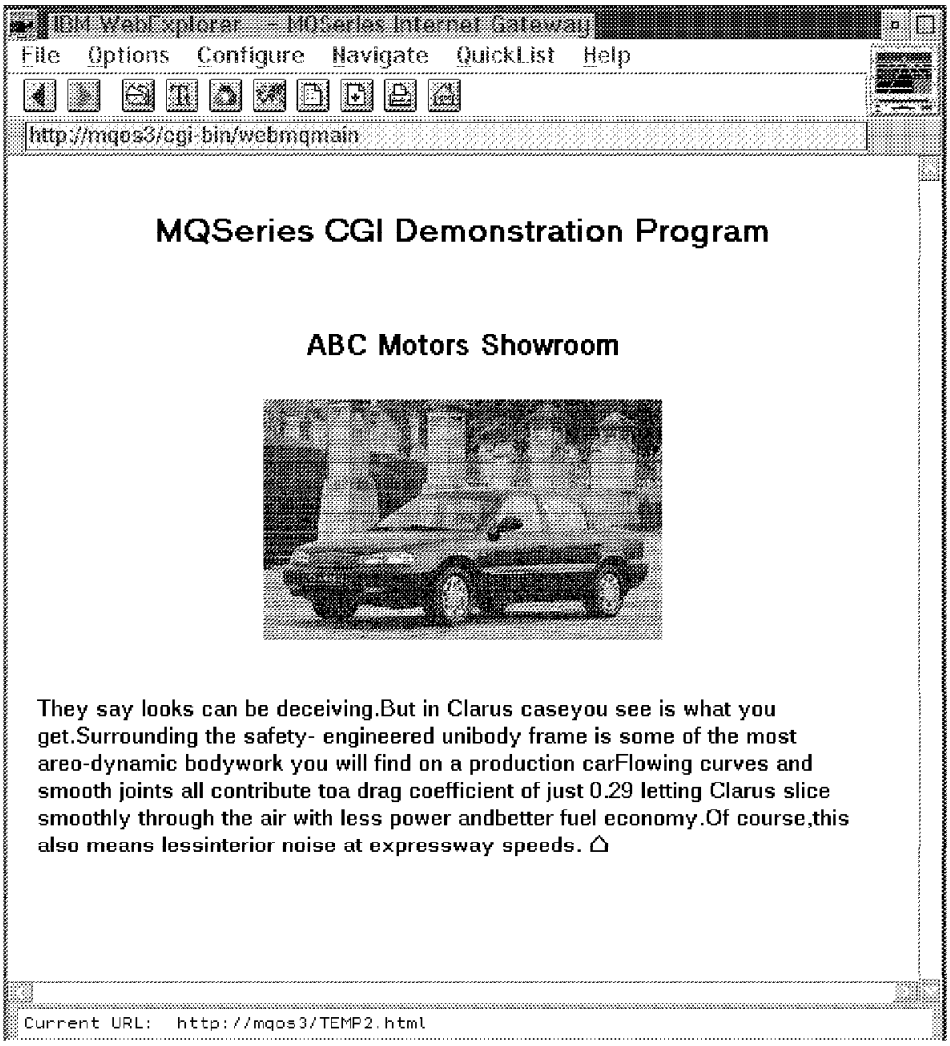


Figure 10. Web Application Result HTML

This screen shows an error situation when the connection to the application servers is not active.

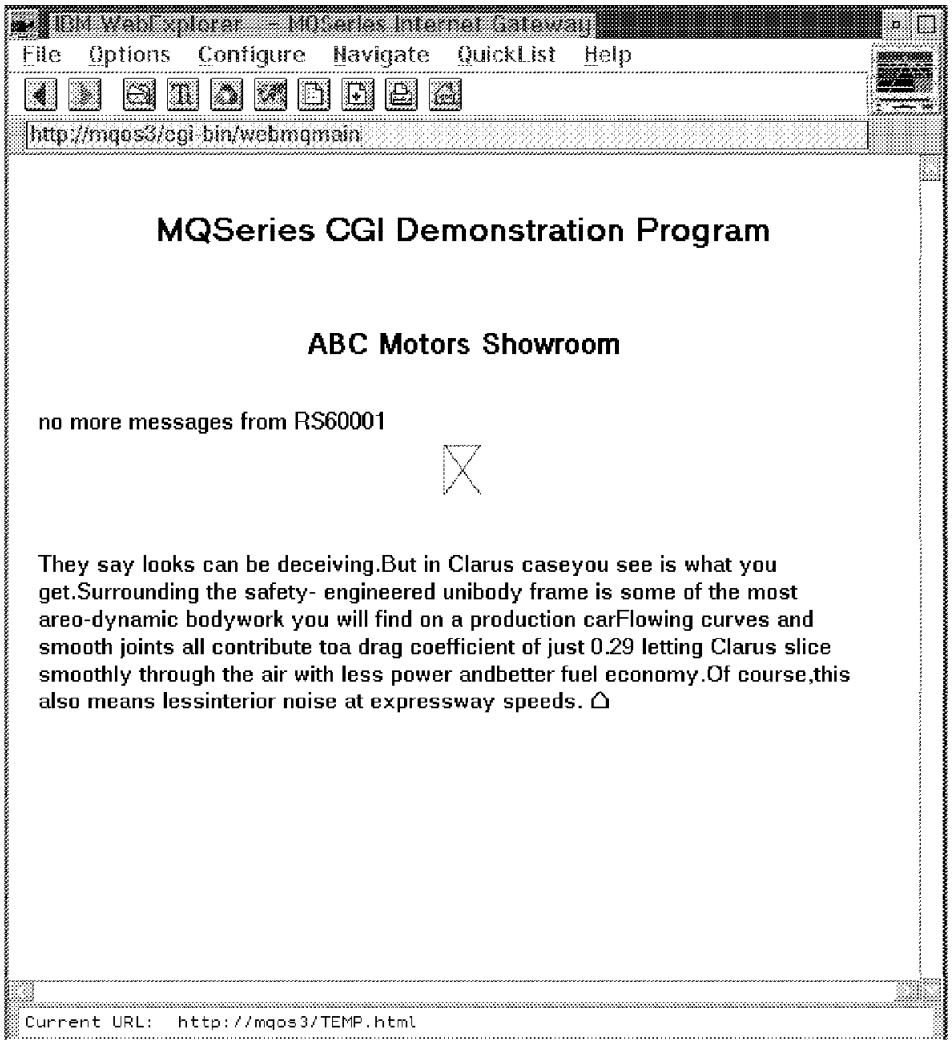


Figure 11. Web Application Result HTML with Error Code

Chapter 4. Example of Using the MQSeries Internet Gateway SupportPac with WTPING/WTPONG

This chapter shows our second example of MQSeries on the Web server. In this example, we used the MQSeries Internet Gateway SupportPac programs as provided, without building our own Web-enabled program.

WTPING and WTPONG are sample MQSeries applications used in ITSO. What the WTPING/WTPONG application basically does is to generate a number of request and reply messages between two MQSeries-connected programs and measures the time of processing the request messages by the server.

In this example we used the MQSeries Internet Gateway SupportPac in its native function. All that had to be done on the Web Server machine was to define a remote queue named WEB.CLIENT.QUEUE into which MQGATE writes the request messages and a local reply queue named Gateway.Reply.Queue.

In this case no user application program is running on the server except the MQSeries Internet Gateway SupportPac program which communicates with our backend servers via MQSeries. In 2.2, “Implementation of a Web-Enabled MQSeries Application” on page 6 we showed an example of using our own CGI application to interface with a WWW server. The MQSeries Internet Gateway SupportPac program provides this function without requiring application coding on the Web server. Either the previously discussed CGI application or the MQSeries Internet Gateway SupportPac are positioned in the spot indicated by ″MQSeries Application″ on the MQSeries server as shown in Figure 2 on page 5.

The MQSeries Internet Gateway SupportPac writes the input parameters which were entered by the Web browser user to the remote WEB.CLIENT.QUEUE. The data progresses through the MQSeries network, is processed by the MQSeries application which sends the reply data back to the Gateway.Reply.Queue. The MQSeries Internet Gateway SupportPac provides the end user with the text data from the reply queue. The MQSeries Internet Gateway SupportPac uses CGI classes (HTML) during all this processing.

The following are the steps we took to run our application.

Before running the application we checked the conditions shown in Figure 12 on page 22. The Internet Connection Server was the server used in this project.

```
OS/2: Is the Internet Connection Server running?
      If not please start it!
      Is the queue manager MQOS3 running ?
      If not please start it!   (strmqm MQOS3)
      Is the channel initiator running ?
      If not please start it!   (runmqchi)
      Is the COMMS-MGR running ?
      If not please start it!

MVS : Is the queue manager CSQ2 running ?
      If not please start it!   (+start qmgr   )
      Is the channel initiator running ?
      We used CICS for DQM function, so check is CICS18 running
      If not please start it!   (s cics18)
      Is the Batch WTPING job running?
      If not please start it!   (submit runjiw1)
      Is the Batch WTPONG job running?
      If not please start it!   (submit runjis8)

Connectivity checklist:
      Do ping channel commands from OS/2 to MVS and vice versa.
```

Figure 12. Setup Checklist. All these steps are checked on the Web server and application server.

After those basic checks, we run the application from the Web browser which uses the resources of the Web server (refer to Figure 1 on page 4).

On the Web browser:

Start IBM WebExplorer under OS/2. Enter Web site http://mqos3/mqping.html.

Enter data in the entry fields or take initial values.

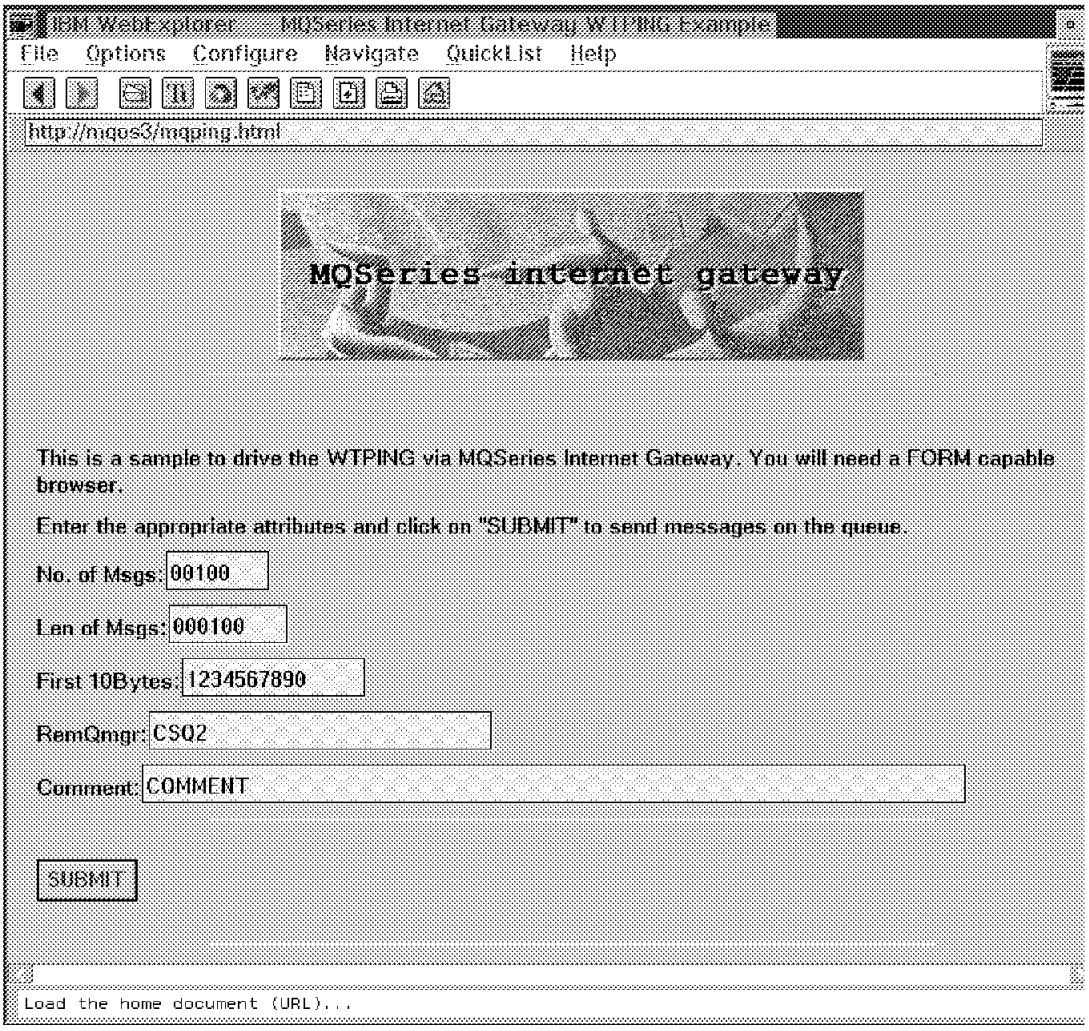


Figure 13. Web Application HTML

Select the **SUBMIT** button after entering the values.

The application runs and presents the following page. It is basically the run-time report which is created by the WTPING batch application on the MVS host system.

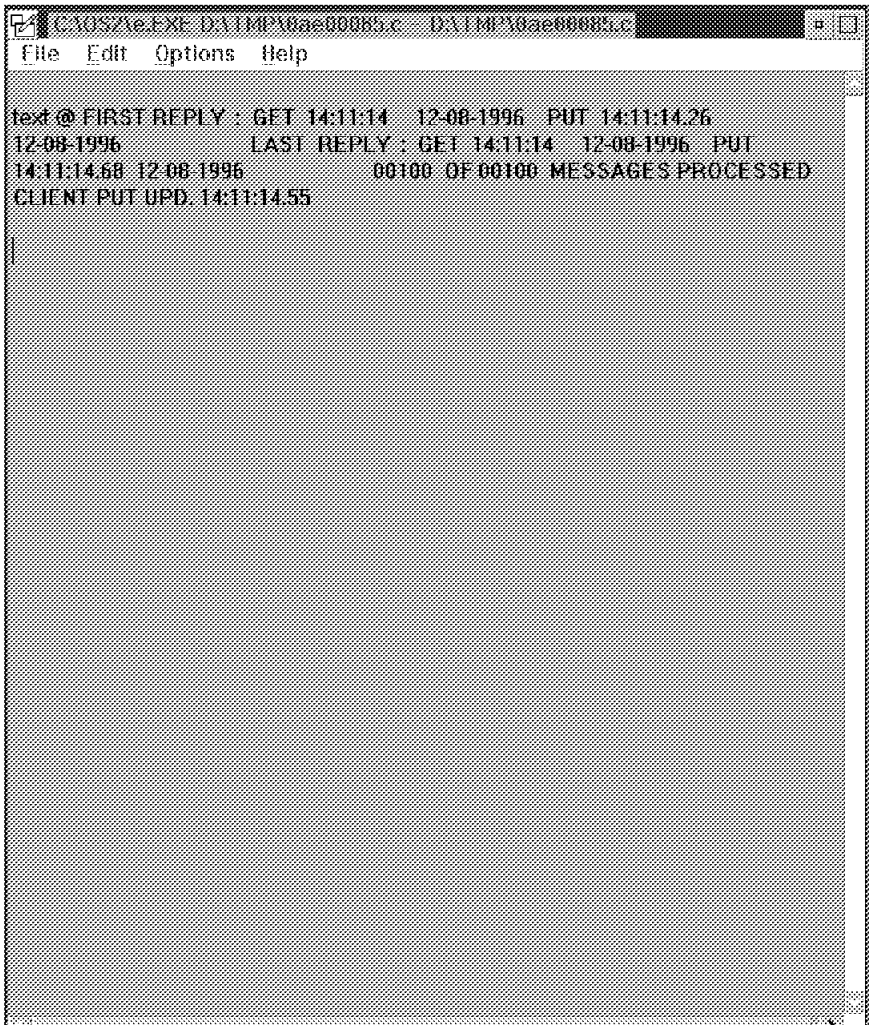


Figure 14. Web Application Result HTML

In this example we have shown how the Internet-connected user can easily be connected to business application servers by using the MQSeries Internet Gateway SupportPac as a message router to applications which are connectable via MQSeries services on other application server platforms.

This allows the end user to make use of MQSeries applications while connected to the Web without having to implement MQSeries at the Web browser machine.

4.1 Web File Browser Sample Using MQSeries Internet Gateway SupportPac

What this application basically does, is access a sequential file on the MVS side and show the content of the file to the end user at the Web browser. This could be an example for showing a log file or sequential output of a host application to a Web user.

This example also used the MQSeries Internet Gateway SupportPac in its native function. On the Web Server, we defined a remote queue WEB.MGMT.QUEUE and continued to use the reply queue Gateway.Reply.Queue.

Again, no user application program was running on the server except the MQSeries Internet Gateway SupportPac program which communicates with our backend server via MQSeries. The following are the steps we took to run our application.

Before running the application we checked the conditions shown in Figure 15 on page 25.

```
OS/2: Is the Internet Connection Server running?
      If not please start it!
      Is the queue manager MQOS3 running ?
      If not please start it!   (strmqm MQOS3)
      Is the channel initiator running ?
      If not please start it!   (runmqchi)
      Is the COMMS-MGR running ?
      If not please start it!

MVS : Is the queue manager CSQ2 running ?
      If not please start it!   (+start qmgr   )
      Is the channel initiator running ?
      We used CICS for DQM function, so check is CICS18 running
      If not please start it!   (s cics18)
      Is the Batch File server job running?
      If not please start it!   (submit runjiw2)

Connectivity checklist:
      Do ping channel commands from OS/2 to MVS and vice versa.
```

Figure 15. Setup Checklist. All these steps are checked on the Web server and application server.

After those basic checks, we run the application from the Web browser which uses the resources of the Web server (refer to Figure 1 on page 4).

On the Web browser:

Start IBM WebExplorer under OS/2. Enter Web site http://mqos3/wtmqcat.html.

Click on the radio button and select an option.

Figure 16. Web Application HTML

Select the **SUBMIT** button after selecting an option.

The application runs and presents the following page. The output shows the content of a sequential file which was accessed by the batch application on the MVS host system.

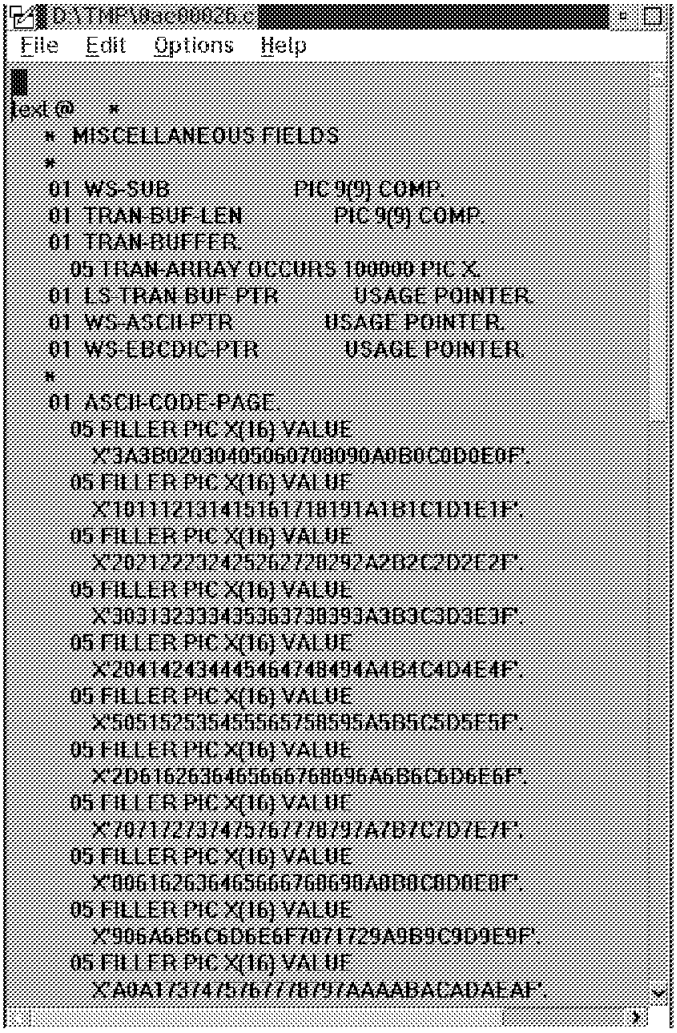


Figure 17. Web Application Result HTML

4.1.1 Comments on HTML with MQSeries Internet Gateway SupportPac

In the previous example discussed in Chapter 3, “Example of MQSeries Web Usage using User Written CGI” on page 15, the output back to the Web browser contained HTML tags as put there by the CGI program.

In this chapter’s examples the output back to the Web browser contained only text data (no HTML tags). The user could have chosen to put HTML tags in the MQSeries message which was returned to the queue named Gateway.Reply.Queue. This is purely a matter of coding (or, in our sample: placing the HTML tags in the host-located file which was sent).

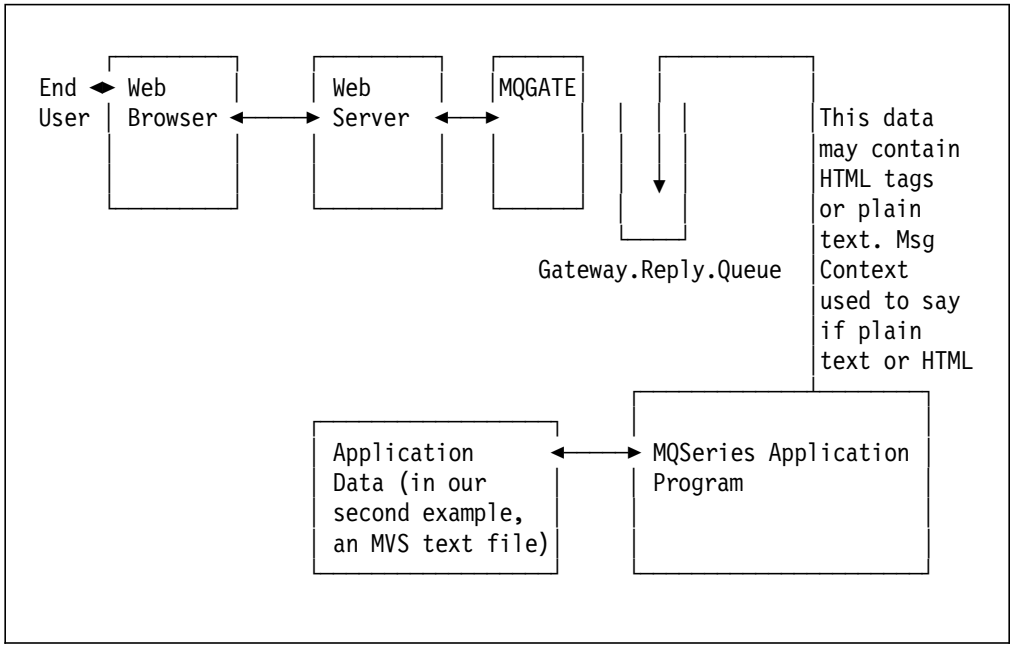


Figure 18. Placing HTML Tags in Reply-to-Browser Message

In our case, we put the HTML-tagged data into the MVS text file and modified the MQSeries message context data sent back to the Gateway.Reply.Queue to indicate HTML is included.

The resultant display at the Web browser is shown in Figure 19 on page 29. The HTML input file is in Figure 48 on page 185.

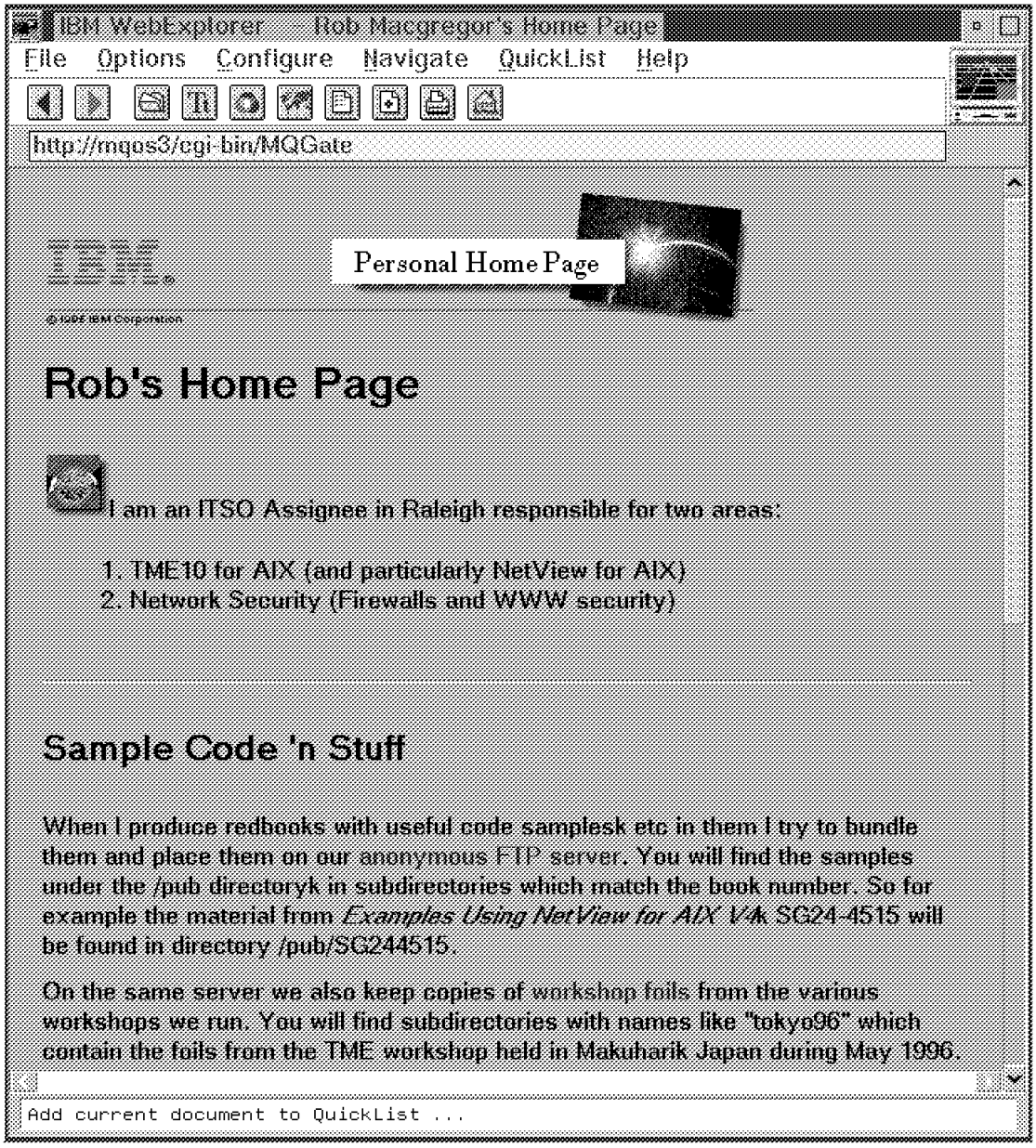


Figure 19. Web Browser Display of S/390-originated index.rob.html File

4.2 Summary

In these examples we have shown how the Internet-connected user can easily be connected to business application servers by using the MQSeries Internet Gateway SupportPac as a message router to applications which are connectable via MQSeries services on other application server platforms.

As with the previously discussed user CGI application (Chapter 3, “Example of MQSeries Web Usage using User Written CGI” on page 15) this allows the end user to make use of MQSeries applications while connected to the Web without having to implement MQSeries at the Web browser machine.

Appendix A. Source Listing and Compile Procedures

Note: All information in this appendix will be available after publication of this document in:

Anonymous FTP server on 9.24.104.108
Directory: /u/ftp/pub/wtmqwww

This appendix contains source listings and procedures for the CICS, OS/2 and AIX work used during this project.

A.1 Cobol CICS Transaction Source Code

```
CBL XOPTS(ANSI85)
CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
      * ------------------------------------------------------------- *
       IDENTIFICATION DIVISION.
      * ------------------------------------------------------------- *
       PROGRAM-ID. PRGQMIW1.
      *REMARKS
      * ------------------------------------------------------------- *
      * Server program for subarea SA18
      * Name of the transaction: W E B   1
      * Name of the program: PRGQM I W   1:
      *                      !     ! !   !
      *                      !     ! !   +---> team name (now fixed)
      *                      !     ! +-------> Server program
      *                      !     !
      *                      !     +---------> S.A. 18
      *                      +---------------> Fixed identifier
      *
      * Input  (request) queue: WEB.REQUESTQ.CSQ2 (fixed)
      * Output (reply)   queue: comes from ReplyToQueue field
      *
      * NOTE: when cloning this program, mind the queue manager name:
      *       they must be changed from CSQ1 to something else
      ****************************************************************
      *   @START_COPYRIGHT@                                          *
      *  Environment           : CICS/ESA Version 3.3; COBOL II      *
      *                                                              *
      *  Function    : This program processes messages from the      *
      *                request queue and put responses to the        *
      *                response queue                                *
      *                                                              *
      * ************************************************************ *
         EJECT
      * ************************************************************ *
      *                                                              *
      *                    Program logic                             *
      *                    -------------                             *
      *                                                              *
      *START.                                                        *
      *    open request queue.                                       *
```

Figure 20 (Part 1 of 19). COBOL Program

```
*    if open fails                                                *
*       write error to tsq and tdq (csml)                         *
*       exit program                                              *
*                                                                 *
*    Get first msg from the request queue.                        *
*    if get fails                                                 *
*       exit the program                                          *
*    Get replytoqueue from message descriptor                     *
*                                                                 *
*    Perform until compcode not = ok                              *
*                     perform ask-datetime                        *
*                     open replytoqueue (exit if error)           *
*                     perform put-respq                           *
*                     close replytoqueue (exit if error)          *
*             End-evaluate                                        *
*             get next msg from request-queue                     *
*    end-perform.                                                 *
*    if unexpected compcode                                       *
*        exit program with error message                          *
*    if no-msg-available                                          *
*       exit program w/o any error message                        *
*    close all queues.                                            *
*    return to cics.                                              *
*                                                                 *
* *************************************************************** *
 EJECT
* ------------------------------------------------------------- *
 ENVIRONMENT DIVISION.
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 DATA DIVISION.
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 WORKING-STORAGE SECTION.
 01  WOO-KEYTABLE.
     03 WOO-KEYS OCCURS 10 INDEXED BY WOO-K1.
       05 WOO-KEYSTRING            PIC X(10).
       05 WOO-KEYDESC              PIC X(1000).
```

Figure 20 (Part 2 of 19). COBOL Program

```
      *
      *   Table for car descriptions to be send to web server
      *
      *
      *
       01  WOO-INITKEY.
           05 FILLER    PIC X(10) VALUE '0000000001'.
           05 FILLER    PIC X(50)
           value 'They say looks can be deceiving.But in Clarus case'.
           05 FILLER    PIC X(50)
           value 'you see is what you get.Surrounding the safety-   '.
           05 FILLER    PIC X(50)
           value 'engineered unibody frame is some of the most areo-'.
           05 FILLER    PIC X(50)
           value 'dynamic bodywork you will find on a production car'.
           05 FILLER    PIC X(50)
           value 'Flowing curves and smooth joints all contribute to'.
           05 FILLER    PIC X(50)
           value 'a drag coefficient of just 0.29 letting Clarus    '.
           05 FILLER    PIC X(50)
           value 'slice smoothly through the air with less power and'.
           05 FILLER    PIC X(50)
           value 'better fuel economy.Of course,this also means less'.
           05 FILLER    PIC X(50)
           value 'interior noise at expressway speeds.              '.
           05 FILLER    PIC X(550) value space.
           05 FILLER    PIC X(10) VALUE '0000000002'.
           05 FILLER    PIC X(50)
           value 'One exceptional aspect of the new Sephia is its   '.
           05 FILLER    PIC X(50)
           value 'attractive body,which suggests something otherthan'.
           05 FILLER    PIC X(50)
           value 'an economy car. The original Sephia was endowed   '.
           05 FILLER    PIC X(50)
           value 'with a sleek form that yielded a drag coefficient '.
           05 FILLER    PIC X(50)
           value 'of just 0.31. Without altering this figure we made'.
           05 FILLER    PIC X(50)
           value 'several exterior changes that make the new Sephia '.
           05 FILLER    PIC X(50)
```

Figure 20 (Part 3 of 19). COBOL Program

```
           value 'even more attractive.TheSephia front and rear have'.
           05 FILLER    PIC X(50)
           value 'been restyled for a cleaner,more aggressive look. '.
           05 FILLER    PIC X(50)
           value 'The hood curves slightly down toward the grill,   '.
           05 FILLER    PIC X(50)
           value 'ascenting the halogen headlights, while the       '.
           05 FILLER    PIC X(50)
           value 'directional lamps have been strategical relocated '.
           05 FILLER    PIC X(50)
           value 'to the corners of the front bumper.Therear receive'.
           05 FILLER    PIC X(50)
           value 'd a larger backing lamp and clear wrap around     '.
           05 FILLER    PIC X(50)
           value 'combination lamps for maximum visibility from     '.
           05 FILLER    PIC X(300) value space.
           05 FILLER    PIC X(10) VALUE '0000000003'.
           05 FILLER    PIC X(50)
           value 'The Avella is a peppy hatchback that sports a bead'.
           05 FILLER    PIC X(50)
           value 'lin extending from thetop of thehood to the grill.'.
           05 FILLER    PIC X(50)
           value 'The wed-saped side panels minimize air resistance '.
           05 FILLER    PIC X(50)
           value 'while the C-post and backseat glass are directly  '.
           05 FILLER    PIC X(50)
           value 'connected, encasing the rear of the vehicle in all'.
           05 FILLER    PIC X(50)
           value 'glass. The soft body curve extends all the way to '.
           05 FILLER    PIC X(50)
           value 'the back a nicely balanced feel.                  '.
           05 FILLER    PIC X(650) value space.
           05 FILLER    PIC X(10) VALUE '0000000004'.
           05 FILLER    PIC X(50)
           value 'The Pride dynamic and sporty silhouette is the    '.
           05 FILLER    PIC X(50)
           value 'result of extensive aerodynamic testing in wind   '.
```

Figure 20 (Part 4 of 19). COBOL Program

```
          05 FILLER    PIC X(50)
          value 'tunnels which smoothed the exterior lines and    '.
          05 FILLER    PIC X(50)
          value 'contours to achieve a very impressive drag       '.
          05 FILLER    PIC X(50)
          value 'coefficient of 0.35.                             '.
          05 FILLER    PIC X(50)
          value 'Design engineers pay extra attention to mximizing'.
          05 FILLER    PIC X(50)
          value 'leg room, headroom and shoulder space.           '.
          05 FILLER    PIC X(650) value space.
          05 FILLER    PIC X(10) VALUE '0000000005'.
          05 FILLER    PIC X(50)
          value 'Sortage is sleek, muscualr looks are the result  '.
          05 FILLER    PIC X(50)
          value 'of extensive wind tunnel testing. The low nose   '.
          05 FILLER    PIC X(50)
          value 'and smooth body contours help pare down the Cd to'.
          05 FILLER    PIC X(50)
          value '0.39, impressive for a car, even more impressive '.
          05 FILLER    PIC X(50)
          value 'for a four-wheel drive.                          '.
          05 FILLER    PIC X(750) value space.
          05 FILLER    PIC X(10) VALUE '0000000006'.
          05 FILLER    PIC X(50)
          value 'Pregio vans sport a sleek,muscularlook that is the'.
          05 FILLER    PIC X(50)
          value 'picture of sophistication. The windshield is     '.
          05 FILLER    PIC X(50)
          value 'positioned at 49-degree slant that gives not only'.
          05 FILLER    PIC X(50)
          value 'the appearance but also the aerodynamics of      '.
          05 FILLER    PIC X(50)
          value 'a passenger car.                                 '.
          05 FILLER    PIC X(750) value space.
          05 FILLER    PIC X(10) VALUE '0000000007'.
          05 FILLER    PIC X(50)
          value 'The sophistication begins at the air dynamically '.
```

Figure 20 (Part 5 of 19). COBOL Program

```
           05 FILLER    PIC X(50)
           value 'raked nose which slices through the air, providing'.
           05 FILLER    PIC X(50)
           value 'exceptional stability and visibility. The besta   '.
           05 FILLER    PIC X(50)
           value 'integrated grill and headlights design give a     '.
           05 FILLER    PIC X(50)
           value 'sophisticated accent to the front end.            '.
           05 FILLER    PIC X(750) value space.
           05 FILLER    PIC X(10) VALUE '0000000008'.
           05 FILLER    PIC X(50)
           value 'The KEV-4 was developed as a commuter car for pepl'.
           05 FILLER    PIC X(50)
           value 'who enjoy the freedom of driving to work. Powered '.
           05 FILLER    PIC X(50)
           value 'mainly by electricity and a 800cccombustion engine'.
           05 FILLER    PIC X(50)
           value 'for auxiliary power, this new concept vehicle     '.
           05 FILLER    PIC X(50)
           value 'meets todays demands for environment friendly     '.
           05 FILLER    PIC X(50)
           value 'transportation.                                   '.
           05 FILLER    PIC X(700) value space.
           05 FILLER    PIC X(10) VALUE '0000000009'.
           05 FILLER    PIC X(1000) value space.
           05 FILLER    PIC X(10) VALUE '0000000010'.
           05 FILLER    PIC X(1000) value space.
       01 W00-BARRIER   PIC X(2000) VALUE SPACE.
      * ------------------------------------------------------------ *
      *
      *    W00 - General work fields
      *
       01  W00-MESSAGE                PIC X(70).
       01  W00-WAIT-INTERVAL          PIC S9(09) BINARY VALUE 60000.
       01  W00-INPUT-MSG-PRIORITY     PIC S9(09) BINARY.
       01  W00-SUB                    PIC S9(09) BINARY.
       01  W00-INDEX                  PIC S9(09) BINARY.
```

Figure 20 (Part 6 of 19). COBOL Program

```
 01  MO1-MESSAGE-CON.
     03 MO1-CON1                 PIC X(18).
     03 MO1-CON2                 PIC X(50).
*
*    Queue names
*
 01  WO2-QUEUE-NAMES.
     05  WO2-REQUEST-QNAME       PIC X(48) VALUE
     'WEB.REQUESTQ.CSQ2                          '.
     05  WO2-RESPONSE-QNAME      PIC X(48) VALUE
     'DUMMY                                      '.
     05  WO2-RESPONSE-QMGR       PIC X(48) VALUE
     'DUMMY                                      '.
*
*    WO3 - MQM API fields
*
 01  WO3-REQ-MSGID               PIC X(24).
 01  WO3-REQ-CORRELID            PIC X(24).
 01  WO3-SELECTORCOUNT           PIC S9(9) BINARY VALUE 1.
 01  WO3-INTATTRCOUNT            PIC S9(9) BINARY VALUE 1.
 01  WO3-CHARATTRLENGTH          PIC S9(9) BINARY VALUE ZERO.
 01  WO3-CHARATTRS               PIC X     VALUE LOW-VALUES.
 01  WO3-HCONN                   PIC S9(9) BINARY VALUE ZERO.
 01  WO3-OPTIONS                 PIC S9(9) BINARY.
 01  WO3-HOBJ-REQUESTQ           PIC S9(9) BINARY.
 01  WO3-HOBJ-RESPQ              PIC S9(9) BINARY.
 01  WO3-HOBJ-DEADQ              PIC S9(9) BINARY.
 01  WO3-COMPCODE                PIC S9(9) BINARY.
 01  WO3-REASON                  PIC S9(9) BINARY.
 01  WO3-SELECTORS-TABLE.
     05  WO3-SELECTORS           PIC S9(9) BINARY OCCURS 2 TIMES.
 01  WO3-INTATTRS-TABLE.
     05  WO3-INTATTRS            PIC S9(9) BINARY OCCURS 2 TIMES.
 01  WO3-DATALEN                 PIC S9(9) BINARY.
 01  WO3-BUFFLEN                 PIC S9(9) BINARY.
 01  GET-COMPCODE                PIC S9(9) BINARY.
 01  GET-REASON                  PIC S9(9) BINARY.
```

Figure 20 (Part 7 of 19). COBOL Program

```
      *
       01  WO3-PUT-BUFFER.
           COPY PRGQO1XA.
       01  WO3-GET-BUFFER.
               05  WO3-GET-KEY             PIC X(10).
      *
      *    TS message for diagnostic trace
      *       Queue name : PRGQMSG
      *
       01  DIAG-MSG                    PIC X(80).
      *
      *
      *    API control blocks
      *
       01  MQM-OBJECT-DESCRIPTOR.
           COPY CMQODV.
       01  MQM-MESSAGE-DESCRIPTOR.
           COPY CMQMDV.
       01  MQM-PUT-MESSAGE-OPTIONS.
           COPY CMQPMOV.
       01  MQM-GET-MESSAGE-OPTIONS.
           COPY CMQGMOV.
       01  MQM-TRIGGER-MESSAGE.
           COPY CMQTML.
      *
      *    CICS ts queue fields
      *
       01  WO5-TD-MESSAGE-LENGTH    PIC S9(4) BINARY.
       01  WO5-ABSTIME              PIC S9(15) COMP-3.
       01  WO5-DATETIME.
           03  WO5-DATE             PIC X(8).
           03  WO5-TIME             PIC X(8).
      *
      *    PRGQO1X8 contains error message to be written to TDQ, TSQ
      *
       COPY PRGQO1X8.
      *
      *    main process flags
      *
```

Figure 20 (Part 8 of 19). COBOL Program

```
 01  WO6-MAIN-PROCESS-FLAG    PIC 9 VALUE 0.
     88 END-PROCESS VALUE 1.
 01  WO6-END-PROCESS          PIC 9 VALUE 1.
*
 01  WO6-INQUIRYQ-STATUS      PIC X(6) VALUE 'CLOSED'.
     88 INQUIRYQ-OPEN   VALUE 'OPEN'.
     88 INQUIRYQ-CLOSED VALUE 'CLOSED'.
*
 01  WO6-CALL-STATUS          PIC X(6) VALUE 'OK'.
     88 CALLS-OK       VALUE 'OK'.
 01  WO6-CALL-ERROR           PIC X(6) VALUE 'FAILED'.
*
 01  WO6-MSG-STATUS           PIC 9 VALUE 0.
     88 MSG-COMPLETE     VALUE 1.
     88 MSG-NOT-COMPLETE VALUE 0.
*
*
*    MQV contains constants (for filling in the control blocks)
*    and return codes (for testing the result of a call)
*
 01  W99-MQV.
 COPY CMQV SUPPRESS.
* ---------------------------------------------------------- *
 PROCEDURE DIVISION.
* ---------------------------------------------------------- *
* ---------------------------------------------------------- *
 A-MAIN SECTION.
     MOVE SPACE TO WOO-KEYTABLE.
     SET WOO-K1 TO 1.
* ---------------------------------------------------------- *
* ---------------------------------------------------------- *
 OPEN-REQUEST-Q.
* ---------------------------------------------------------- *
*                                                            *
* This section opens the request queue for input shared      *
*                                                            *
* ---------------------------------------------------------- *
*
```

Figure 20 (Part 9 of 19). COBOL Program

```
           MOVE MQOT-Q            TO MQOD-OBJECTTYPE.
           MOVE WO2-REQUEST-QNAME TO MQOD-OBJECTNAME.
      *
           COMPUTE WO3-OPTIONS    =  MQOO-INPUT-SHARED +
                                     MQOO-SAVE-ALL-CONTEXT.
           MOVE ZERO TO WO3-HCONN.
      *
           CALL 'MQOPEN' USING WO3-HCONN
                               MQOD
                               WO3-OPTIONS
                               WO3-HOBJ-REQUESTQ
                               WO3-COMPCODE
                               WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'              TO MO2-OPERATION
              MOVE WO2-REQUEST-QNAME     TO MO2-OBJECTNAME
              PERFORM RECORD-CALL-ERROR
              GO   TO A-MAIN-EXIT
           END-IF
      *
      * Program identifies itself to the system log
      *
           MOVE 'MQS95001 : 95001 SA25 CSQ1 WEB1 STARTED'
           TO DIAG-MSG
           EXEC CICS WRITE OPERATOR TEXT(DIAG-MSG)
           END-EXEC.
      *
      *
       MAIN-PROCESS.
      *-------------------------------------------------------------- *
      *    Loop until messages are in the queue
      *-------------------------------------------------------------- *
      *
      *    Initialize the Get Message Options (MQGMO) control block.
      *    (The copy book initializes the remaining fields)
      *
```

Figure 20 (Part 10 of 19). COBOL Program

```
           COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                                   MQGMO-ACCEPT-TRUNCATED-MSG +
                                   MQGMO-NO-SYNCPOINT.
           MOVE WO0-WAIT-INTERVAL   TO MQGMO-WAITINTERVAL
      *
           MOVE LENGTH OF WO3-GET-BUFFER TO WO3-BUFFLEN.
           MOVE MQMI-NONE                TO MQMD-MSGID.
           MOVE MQCI-NONE                TO MQMD-CORRELID.
      *
      *    Make the first MQGET call outside the loop
      *
           CALL 'MQGET' USING WO3-HCONN
                              WO3-HOBJ-REQUESTQ
                              MQMD
                              MQGMO
                              WO3-BUFFLEN
                              WO3-GET-BUFFER
                              WO3-DATALEN
                              GET-COMPCODE
                              GET-REASON.
      *

            IF GET-COMPCODE = MQCC-FAILED
              MOVE 'MQS95001 : 95001 SA25 CSQ1 MQGET ERROR '
              TO DIAG-MSG
              EXEC CICS WRITE OPERATOR TEXT(DIAG-MSG)
              END-EXEC
              MOVE 'MQGET FIRST '    TO MO2-OPERATION
              MOVE WO2-REQUEST-QNAME TO MO2-OBJECTNAME
              MOVE GET-COMPCODE  TO WO3-COMPCODE
              MOVE GET-REASON    TO WO3-REASON
              PERFORM RECORD-CALL-ERROR
              GO TO A-MAIN-EXIT
            END-IF
      *
      *
      *    Loop from here to END-PERFORM until the MQGET call fails
      *
```

Figure 20 (Part 11 of 19). COBOL Program

```
           PERFORM WITH TEST BEFORE
                   UNTIL GET-COMPCODE = MQCC-FAILED
              MOVE MQMD-MSGID     TO  WO3-REQ-MSGID
              MOVE MQMD-CORRELID  TO  WO3-REQ-CORRELID
      *
      *       Save ReplyToQueue name
      *
              MOVE MQMD-REPLYTOQ TO WO2-RESPONSE-QNAME
              MOVE MQMD-REPLYTOQMGR TO WO2-RESPONSE-QMGR
           MOVE WO3-GET-BUFFER TO DIAG-MSG
           EXEC CICS WRITE OPERATOR TEXT(DIAG-MSG)
           END-EXEC
      *
      *
              PERFORM OPEN-RESPONSE-Q
              PERFORM PUT-RESPONSE-MESSAGE
              PERFORM CLOSE-RESPONSE-Q
      *
      *       Clear MQMD-MSGID and MQMD-CORRELID before the next
      *       MQGET call to ensure that all messages are retrieved
      *
              MOVE MQMI-NONE TO MQMD-MSGID
              MOVE MQCI-NONE TO MQMD-CORRELID
      *
      *       Get the next message
      *
              CALL 'MQGET' USING WO3-HCONN
                                 WO3-HOBJ-REQUESTQ
                                 MQMD
                                 MQGMO
                                 WO3-BUFFLEN
                                 WO3-GET-BUFFER
                                 WO3-DATALEN
                                 GET-COMPCODE
                                 GET-REASON
      *
      *
      *       Test the output of the MQGET call at the top of the loop.
      *       Exit the loop if an error occurs
      *
           END-PERFORM.
```

Figure 20 (Part 12 of 19). COBOL Program

```
*
*    Test the output of the MQGET call.  If the call failed,
*    print an error message showing the completion code and
*    reason code, unless the reason code is NO-MSG-AVAILABLE.
*
*    Note: When the loop reaches the end of the file, the
*          completion code is MQCC-FAILED and the reason code
*          is MQRC-NO-MSG-AVAILABLE. If this happens, the
*          program closes with no error message
*
     IF GET-REASON   NOT = MQRC-NO-MSG-AVAILABLE
        MOVE 'MQGET NEXT  '     TO MO2-OPERATION
        MOVE WO2-REQUEST-QNAME TO MO2-OBJECTNAME
        MOVE GET-COMPCODE      TO WO3-COMPCODE
        MOVE GET-REASON        TO WO3-REASON
        PERFORM RECORD-CALL-ERROR
     END-IF.
*
*
 CLOSE-ALL.
*
     PERFORM CLOSE-REQUEST-Q.
*
 A-MAIN-EXIT.
     EXEC CICS RETURN
     END-EXEC.
*
     GOBACK.
     EJECT
*
 OPEN-RESPONSE-Q SECTION.
* ------------------------------------------------------------ *
*                                                              *
* This section opens the response queue for output             *
* The response queue name to open is in WO2-RESPONSE-QNAME     *
*                                                              *
* ------------------------------------------------------------ *
*
     MOVE MQOT-Q             TO MQOD-OBJECTTYPE.
```

Figure 20 (Part 13 of 19). COBOL Program

```
           MOVE WO2-RESPONSE-QNAME TO MQOD-OBJECTNAME.
           MOVE WO2-RESPONSE-QMGR  TO MQOD-OBJECTQMGRNAME.
      *
           COMPUTE WO3-OPTIONS    =  MQOO-OUTPUT.
      *
           CALL 'MQOPEN' USING WO3-HCONN
                               MQOD
                               WO3-OPTIONS
                               WO3-HOBJ-RESPQ
                               WO3-COMPCODE
                               WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
               MOVE 'MQOPEN'             TO MO2-OPERATION
               MOVE WO2-RESPONSE-QNAME   TO MO2-OBJECTNAME
               PERFORM RECORD-CALL-ERROR
               GO   TO A-MAIN-EXIT
           END-IF.
      *
       OPEN-RESPONSE-Q-EXIT.
           EXIT.
           EJECT
      *
      * ------------------------------------------------------------ *
       CLOSE-REQUEST-Q SECTION.
      * ------------------------------------------------------------ *
      *                                                              *
      * This section closes the request queue                        *
      *                                                              *
      * ------------------------------------------------------------ *
      *
           CALL 'MQCLOSE' USING WO3-HCONN
                                WO3-HOBJ-REQUESTQ
                                MQCO-NONE
                                WO3-COMPCODE
                                WO3-REASON.
      *
```

Figure 20 (Part 14 of 19). COBOL Program

```
           IF W03-COMPCODE NOT = MQCC-OK
               MOVE 'MQCLOSE'            TO M02-OPERATION
               MOVE W02-REQUEST-QNAME    TO M02-OBJECTNAME
               PERFORM RECORD-CALL-ERROR
               GO TO A-MAIN-EXIT
           END-IF
      *
      * Program sends a close message to the system log
      *
           MOVE 'MQS95002 : 95002 SA25 CSQ1 WEB ENDED'
           TO DIAG-MSG
           EXEC CICS WRITE OPERATOR TEXT(DIAG-MSG)
           END-EXEC.
      *
       CLOSE-REQUEST-Q-EXIT.
      *
      *    Return to performing section
      *
           EXIT.
           EJECT
      *
      * ------------------------------------------------------------ *
       CLOSE-RESPONSE-Q SECTION.
      * ------------------------------------------------------------ *
      *                                                              *
      * This section closes the response queue                       *
      *                                                              *
      * ------------------------------------------------------------ *
      *
           CALL 'MQCLOSE' USING W03-HCONN
                                W03-HOBJ-RESPQ
                                MQCO-NONE
                                W03-COMPCODE
                                W03-REASON.
      *
           IF W03-COMPCODE NOT = MQCC-OK
               MOVE 'MQCLOSE'            TO M02-OPERATION
```

Figure 20 (Part 15 of 19). COBOL Program

```
            MOVE WO2-RESPONSE-QNAME   TO MO2-OBJECTNAME
            PERFORM RECORD-CALL-ERROR
            GO TO A-MAIN-EXIT
        END-IF.
*
*
 CLOSE-RESPONSE-Q-EXIT.
*
*    Return to performing section
*
        EXIT.
        EJECT
*
* ------------------------------------------------------------- *
 RECORD-CALL-ERROR SECTION.
* ------------------------------------------------------------- *
*                                                               *
* This section writes an error message to the CICS td queue    *
* 'CSML'                                                        *
* The failing operation and object name fields are completed   *
* by the calling application. The remaining fields of the      *
* message are completed by this routine                         *
*                                                               *
* ------------------------------------------------------------- *
*
        EXEC CICS ASKTIME
            ABSTIME(WO5-ABSTIME)
        END-EXEC.
        EXEC CICS FORMATTIME
            ABSTIME(WO5-ABSTIME)
            DDMMYY(MO2-DATE) DATESEP
            TIME(MO2-TIME) TIMESEP
        END-EXEC.
*
        MOVE EIBTRNID        TO MO2-TRANSACTION
                                MO3-TRANSACTION.
        MOVE EIBTASKN        TO MO2-TASK-NUMBER
```

Figure 20 (Part 16 of 19). COBOL Program

```
                              MO3-TASK-NUMBER.
     MOVE WO3-COMPCODE    TO MO2-COMPCODE
     MOVE WO3-REASON      TO MO2-REASON
     MOVE MO2-DATE        TO MO3-DATE.
     MOVE MO2-TIME        TO MO3-TIME.
     MOVE LENGTH OF MO3-CSML-ERROR-MSG
                          TO WO5-TD-MESSAGE-LENGTH.
*
     EXEC CICS WRITEQ TD
         QUEUE('CSML')
         FROM (MO3-CSML-ERROR-MSG)
         LENGTH(WO5-TD-MESSAGE-LENGTH)
     END-EXEC.
*
 RECORD-CALL-ERROR-EXIT.
*
*    Return to performing section
*
     EXIT.
*
*    Return to performing section
*
     EXIT.
     EJECT
*
* ------------------------------------------------------------ *
*
* ------------------------------------------------------------ *
 PUT-RESPONSE-MESSAGE SECTION.
* ------------------------------------------------------------ *
*                                                              *
* Set the object descriptor, message descriptor and put message*
*     options to the values required for creating a output     *
*     message                                                  *
*                                                              *
* ------------------------------------------------------------ *
*
```

Figure 20 (Part 17 of 19). COBOL Program

```
      *
           MOVE 'BUILD  ANSWER MSG' TO DIAG-MSG.
           EXEC CICS WRITE OPERATOR TEXT(DIAG-MSG)
           END-EXEC.
           MOVE MQMT-REPLY         TO  MQMD-MSGTYPE.
           MOVE MQRO-NONE          TO  MQMD-REPORT.
           MOVE SPACES             TO  MQMD-REPLYTOQ.
           MOVE SPACES             TO  MQMD-REPLYTOQMGR.
           MOVE WO3-REQ-MSGID      TO  MQMD-MSGID.
           MOVE WO3-REQ-CORRELID   TO  MQMD-CORRELID.
      *
           COMPUTE MQPMO-OPTIONS  =  MQPMO-NO-SYNCPOINT.
           MOVE WO3-HOBJ-REQUESTQ TO MQPMO-CONTEXT.
      *
      *    enable mqgmo_convert
      *
           MOVE MQFMT-STRING TO MQMD-FORMAT.
      *
           MOVE LENGTH OF WO3-PUT-BUFFER TO WO3-BUFFLEN.
           MOVE WO3-GET-KEY TO WO3-KEY.
           PERFORM SEARCH-KEY.
      *
           CALL 'MQPUT' USING WO3-HCONN
                              WO3-HOBJ-RESPQ
                              MQMD
                              MQPMO
                              WO3-BUFFLEN
                              WO3-PUT-BUFFER
                              WO3-COMPCODE
                              WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQPUT'          TO  MO2-OPERATION
              MOVE MQOD-OBJECTNAME  TO  MO2-OBJECTNAME
              PERFORM RECORD-CALL-ERROR
              GO   TO CLOSE-ALL
           END-IF.
```

Figure 20 (Part 18 of 19). COBOL Program

```
*
 PUT-RESPONSE-MESSAGE-EXIT.
     EXIT.
     EJECT
*
*    Search for the correct description
*    for passed key value
*
*
 SEARCH-KEY SECTION.
     MOVE W00-INITKEY TO W00-KEYTABLE.
     MOVE SPACE TO W03-ERR-MSG.
 SEARCH-START.
     SET W00-K1 TO 1.
 SEARCH-CALL.
     SEARCH W00-KEYS VARYING W00-K1
       AT END MOVE SPACE TO W03-DESCRIPTION
              MOVE 'KEY INVALID' TO W03-ERR-MSG
       WHEN W00-KEYSTRING(W00-K1) = W03-KEY
            MOVE W00-KEYDESC(W00-K1) TO W03-DESCRIPTION
     END-SEARCH.
     MOVE 80 TO W05-TD-MESSAGE-LENGTH.
     MOVE W03-KEY TO DIAG-MSG.
     EXEC CICS WRITEQ TD
         QUEUE('CSML')
         FROM (DIAG-MSG)
         LENGTH(W05-TD-MESSAGE-LENGTH)
     END-EXEC.
     MOVE W03-DESCRIPTION TO DIAG-MSG.
     EXEC CICS WRITEQ TD
         QUEUE('CSML')
         FROM (DIAG-MSG)
         LENGTH(W05-TD-MESSAGE-LENGTH)
     END-EXEC.
 SEARCH-KEY-EXIT.
     EXIT.
* ------------------------------------------------------------- *
*                    END OF PROGRAM                             *
* ------------------------------------------------------------- *
*
```

Figure 20 (Part 19 of 19). COBOL Program

A.2 Cobol COPY PRGQO1X8

This a cobol copy needed for the CICS application server program.

```
       01  MO2-CALL-ERROR-MSG.
           05  MO2-TRANSACTION      PIC X(04).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-TASK-NUMBER      PIC 9(09).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-DATE             PIC X(08).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-TIME             PIC X(08).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-OPERATION.
               10  MO2-OPER-MSG     PIC X(15).
               10                   PIC X(01) VALUE SPACE.
               10  MO2-OPER-MSGID   PIC X(8).
               10                   PIC X(01) VALUE SPACE.
               10  MO2-OPER-CORRID  PIC X(24).
           05                       PIC X(18) VALUE
               ' ERROR - COMPCODE:'.
           05  MO2-COMPCODE         PIC Z(08)9.
           05                       PIC X(13) VALUE
               ' REASON CODE:'.
           05  MO2-REASON           PIC Z(08)9.
           05                       PIC X(12) VALUE
               ' OBJECTNAME:'.
           05  MO2-OBJECTNAME       PIC X(48).
       01  MO3-CSML-ERROR-MSG.
           05                       PIC X(37) VALUE
             'An error has occurred in transaction '.
           05  MO3-TRANSACTION      PIC X(04).
           05                       PIC X(10) VALUE
             ' Task no. '.
           05  MO3-TASK-NUMBER      PIC 9(07).
           05                       PIC X(01) VALUE SPACE.
           05  MO3-DATE             PIC X(08).
           05                       PIC X(01) VALUE SPACE.
           05  MO3-TIME             PIC X(08).
           05                       PIC X(40) VALUE
             ' View temporary storage queue ''PRGQO1X'''.
       01  MO4-STARTUP-ERROR        PIC X(60) VALUE
             ' started without data, a MQTM structure was expected.'.
```

Figure 21. CICS COBOL Copy PRGQO1X8

A.3 Cobol COPY PRGQO1XA

This is a cobol copy needed for the CICS application server program.

```
      *    REPLY   QUEUE MESSAGE BUFFER FORMAT
           03  WO3-REQUEST.
               05  WO3-KEY                 PIC X(10).
               05  WO3-DESCRIPTION         PIC X(1000).
               05  WO3-ERR-MSG             PIC X(80).
```

Figure 22. CICS COBOL Copy PRGQO1XA

A.4 Cobol CICS Compile

This compile procedure was used to compile the CICS application server.

```
//COMPPRG JOB 'MQ SAMPLE',MSGCLASS=6,CLASS=I,NOTIFY=WTWKSH4
//*-------------------------------------------------------------------*
//*
//*  COMPILE CICS COBOL APPLICATIONS
//*
//*
//*
//*      THIS JOB CONTAINS 4 STEPS
//*      1.   EXEC THE COBOL TRANSLATOR
//*           (USING THE SUPPLIED SUFFIX 1$)
//*      2.   EXEC THE VS COBOL II COMPILER
//*      3.   REBLOCK SDFHCOB(DFHEILIC) FOR USE BY THE LINKEDIT STEP
//*      4.   LINKEDIT THE OUTPUT TO CICS.USER.SDFJLOAD
//*
//*      THE FOLLOWING JCL SHOULD BE USED TO EXECUTE THIS PROC
//*
//*      //APPLPROG EXEC DFHEITVL
//*        ....
//*      //TRN.SYSIN  DD *
//*         .
//*         . APPLICATION PROGRAM
//*         .
//*      /*
//*      //*   CONCATENATE THE MQM MVS/ESA LIB WITH SYSLIB DD
//*      //COB.SYSLIB DD ...
//*      //           DD DSN=MQM.V1R3M0.SCSQCOBC,DISP=SHR
//*      //           DD ...
//*      // ....
//*      //* INCLUDE MQM MVS/ESA LIBRARY CONTAINING CICS STUB
//*      //CSQSTUB    DD DSN=MQM.V1R3M0.SCSQLOAD,DIS=SHR
//*      //LKED.SYSIN DD *
//*         INCLUDE CSQSTUB(CSQCSTUB)
//*         ....
//*         NAME ANYNAME(R)
//*      /*
//*
//*      WHERE ANYNAME IS THE NAME OF YOUR APPLICATION PROGRAM.
//*      (REFER TO THE SYSTEM DEFINITION GUIDE FOR FULL DETAILS,
//*      INCLUDING WHAT TO DO IF YOUR PROGRAM CONTAINS CALLS TO
//*      THE COMMON PROGRAMMING INTERFACE.)
//*-------------------------------------------------------------------*
```

Figure 23 (Part 1 of 3). CICS COBOL Compile

```
//*
//DFHEITVL PROC SUFFIX=1$,
//       INDEX='CICS',
//       INDEX2='CICS',
//       OUTC='*',
//       PGMMEM=' ',
//       REG=2048K,
//       LNKPARM='LIST,XREF',
//       WORK=SYSDA
//*
//*
//TRN    EXEC PGM=DFHECP&SUFFIX,
//            PARM='COBOL2',
//            REGION=&REG
//STEPLIB  DD DSN=&INDEX2..SDFHLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=&OUTC
//SYSPUNCH DD DSN=&&SYSCIN,
//            DISP=(,PASS),UNIT=&WORK,
//            DCB=BLKSIZE=400,
//            SPACE=(400,(400,100))
//*
//COB    EXEC PGM=IGYCRCTL,REGION=&REG,
//       PARM='NODYNAM,LIB,OBJECT,RENT,RES,APOST,MAP,XREF'
//STEPLIB  DD DSN=COB2.COB2COMP,DISP=SHR
//SYSLIB   DD DSN=&INDEX..SDFHCOB,DISP=SHR
//         DD DSN=&INDEX..SDFHMAC,DISP=SHR
//         DD DSN=MQM.V1R3M0.SCSQCOBC,DISP=SHR
//         DD DSN=MQM.USER.SOURCE,DISP=SHR        /* COBOL COPYBOOKS */
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DSN=&&SYSCIN,DISP=(OLD,DELETE)
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),
//            UNIT=&WORK,SPACE=(80,(250,100))
//SYSUT1   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT2   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT3   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT4   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT6   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT7   DD UNIT=&WORK,SPACE=(460,(350,100))
```

Figure 23 (Part 2 of 3). CICS COBOL Compile

```
//*
//COPYLINK EXEC PGM=IEBGENER,COND=(7,LT,COB)
//SYSUT1   DD DSN=&INDEX..SDFHCOB(DFHEILIC),DISP=SHR
//SYSUT2   DD DSN=&&COPYLINK,DISP=(NEW,PASS),
//            DCB=(LRECL=80,BLKSIZE=400,RECFM=FB),
//            UNIT=&WORK,SPACE=(400,(20,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DUMMY
//*
//LKED   EXEC PGM=IEWL,REGION=&REG,
//            PARM='&LNKPARM',COND=(5,LT,COB)
//SYSLIB   DD DSN=&INDEX2..SDFHLOAD,DISP=SHR
//         DD DSN=COB2.COB2CICS,DISP=SHR
//         DD DSN=COB2.COB2LIB,DISP=SHR
//         DD DSN=MQM.USER.LOAD,DISP=SHR
//CSQSTUB  DD DSN=MQM.V1R3M0.SCSQLOAD,DISP=SHR
//SYSLMOD  DD DSN=CICS.USER.SDFHLOAD(&PGMMEM),DISP=SHR
//SYSUT1   DD UNIT=&WORK,DCB=BLKSIZE=1024,
//            SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSLIN   DD DSN=&&COPYLINK,DISP=(OLD,DELETE)
//         DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD DDNAME=SYSIN
// PEND
//STEP1    EXEC DFHEITVL,
//         PGMMEM=PRGQMIW1
//TRN.SYSIN DD DSN=MQM.USER.SOURCE(PRGQMIW1),DISP=SHR
//LKED.SYSIN DD *
   INCLUDE CSQSTUB(CSQCSTUB)
   NAME    PRGQMIW1(R)
/*
//
```

Figure 23 (Part 3 of 3). CICS COBOL Compile

A.5 CEDA SCREENS

These are the screen images on how to define CICS tranid and programs.

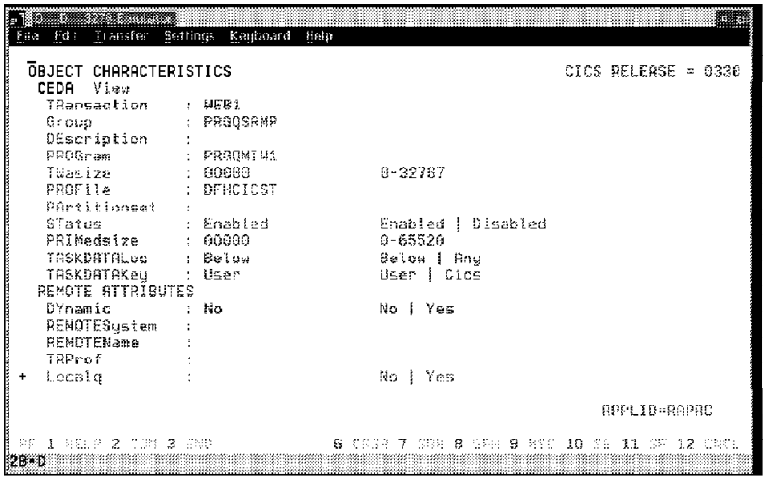


Figure 24. CICS CEDA for Transaction WEB1

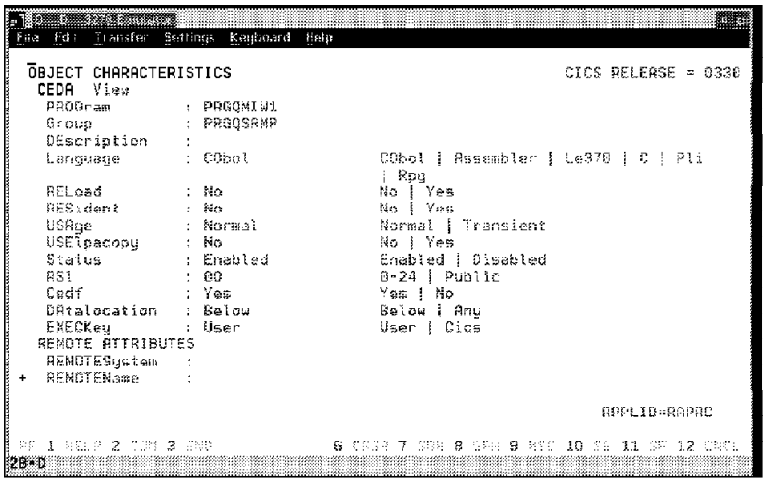


Figure 25. CICS CEDA for Program PRGQMIW1

A.6 OS/2 MAIN Program

This is the Web-enabled MQSeries program.

```
/*********************************************************************/
/*                                                                   */
/* MODULE NAME      webmqmain.cpp                                    */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
/* includes for MQI */
#include <cmqc.h>
#include "CGIPart.h"
#include "CGIPartSet.h"
#include <stream.h>
#include "URLDecoder.h"
#ifdef __OS2__
#include <os2.h>
#endif



int main(int argc, char **argv)
{

   /*   Declare file and character for sample input                 */
   FILE *fp;
   int    i, j;  /* auxiliary counter  */

   /*   Declare MQI structures needed                               */
   MQOD     od = {MQOD_DEFAULT};    /* Object Descriptor            */
   MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor           */
   MQMD     mdDefault = {MQMD_DEFAULT};    /* Message Descriptor      */
   MQPMO    pmo = {MQPMO_DEFAULT};   /* put message options          */
   MQGMO    gmo = {MQGMO_DEFAULT};   /* get message options          */
      /** note, sample uses defaults where it can **/

   MQHCONN  Hcon;                   /* connection handle            */
   MQHOBJ   Hobj;                   /* object handle                */
```

Figure 26 (Part 1 of 17). Web-Enabled MQSeries Program webmqmain

```
MQLONG   O_options;              /* MQOPEN options                */
MQLONG   C_options;              /* MQCLOSE options               */
MQLONG   CompCode;               /* completion code               */
MQLONG   OpenCode;               /* MQOPEN completion code        */
MQLONG   Reason;                 /* reason code                   */
MQLONG   CReason;                /* reason code for MQCONN        */
MQLONG   buflen;                 /* buffer length                 */
MQLONG   messlen;                /* message length                */
char     buffer[1200];           /* message buffer                */
char     Key[11];                /* Key to access                 */
char     QMName[50]="MQOS3";     /* QMGR name                     */

char     tempstr[24];


MQCHAR48 replyQ="WEB.REPLYQ";
MQCHAR48 replyQMgr="MQOS3";

MQBYTE24 saveMsgId;
MQBYTE24 saveCorrelId;

MQBYTE24 saveMsgId1;
MQBYTE24 saveCorrelId1;
MQBYTE24 initmsg;


unsigned int contentLength = 0;
long bLocalReturnCode = TRUE;
CGIPartSet the_CGIPartSet;
unsigned long ulNameLength;
char *messageBuffer;
URLDecoder theDecoder;

char filename[255];


if(strcmp(getenv("REQUEST_METHOD"),"POST")) {
```

Figure 26 (Part 2 of 17). Web-Enabled MQSeries Program webmqmain

```
        printf("This script should be referenced with a METHOD of POST.\n");
        printf("if you don't understand this, see this");
        printf("<A HREF=\"http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/\
               fill-out-forms/overview.html\">forms overview</A>.%c",10);
        exit(1);
    }

    if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded")) {
        printf("This script can only be used to decode form results.\n");
        exit(1);
    }




    contentLength = atoi(getenv("CONTENT_LENGTH"));
    messageBuffer = new char[contentLength +1];

    cin.read(messageBuffer, contentLength);
    messageBuffer[cin.gcount()] = '\0';

    the_CGIPartSet.Initialize(messageBuffer, contentLength);

    ulNameLength = 11;
    bLocalReturnCode = the_CGIPartSet.getPartValue("Key",
                                                   Key,
                                                   &ulNameLength);


/*
printf("<p>Key=%s\n\n", Key);
*/
```

Figure 26 (Part 3 of 17). Web-Enabled MQSeries Program webmqmain

```
   /* HTML */
   printf("Content-type: text/html\n\n");
   printf("<HTML><HEAD><TITLE>MQSeries CGI Program</TITLE></HEAD>\n");
   printf("<BODY BGCOLOR=\"#EOEOFF\">\n");
   /* check key range */
   if ((atoi(Key) <= 0)||(atoi(Key)>=9))
   {
      printf("<P>Invalid key.\n");
      printf("<P>Please input the key again.\n");
      printf("</BODY></HTML>\n");
      delete [] messageBuffer;
      return(0);
   } else
   {
      printf("<center><H2>MQSeries CGI Demonstration Program
      </H2></center>\n\n\n");
      printf("<center><P><H3>ABC Motors Showroom</H3></P></center>\n\n\n");
   }



   if (bLocalReturnCode == FALSE) {
      // No Key...
      Key[0] = 0;      /* default */
   } /* endif */



/*
printf("<p>MQSeries CGI Program Started\n\n");
*/



   /*****************************************************************/
```

Figure 26 (Part 4 of 17). Web-Enabled MQSeries Program webmqmain

```
   /*                                                                  */
   /*   Connect to queue manager                                       */
   /*                                                                  */
   /********************************************************************/
   MQCONN(QMName,                  /* queue manager                    */
          &Hcon,                   /* connection handle                */
          &CompCode,               /* completion code                  */
          &CReason);               /* reason code                      */

   /* report reason and stop if it failed     */
   if (CompCode == MQCC_FAILED)
   {
     printf("<p>MQCONN ended with reason code %ld</p>\n", CReason);
     exit(CReason);
   }

   /********************************************************************/
   /*                                                                  */
   /*   Use parameter as the name of the target queue                  */
   /*                                                                  */
   /********************************************************************/
   strcpy(od.ObjectName, "WEB.REQUESTQ.RS60001");

/*
printf("<p>target queue is %s\n", od.ObjectName);
*/


   /********************************************************************/
   /*                                                                  */
   /*   Open the target message queue for output                       */
   /*                                                                  */
   /********************************************************************/
   O_options = MQOO_OUTPUT           /* open queue for output        */
          + MQOO_FAIL_IF_QUIESCING; /* but not if MQM stopping      */
   MQOPEN(Hcon,                      /* connection handle            */
          &od,                       /* object descriptor for queue  */
          O_options,                 /* open options                 */
          &Hobj,                     /* object handle                */
          &OpenCode,                 /* MQOPEN completion code       */
          &Reason);                  /* reason code                  */
```

Figure 26 (Part 5 of 17). Web-Enabled MQSeries Program webmqmain

```
/* report reason, if any; stop if failed      */
if (Reason != MQRC_NONE)
{
  printf("<p>MQOPEN ended with reason code %ld</p>\n", Reason);
}

if (OpenCode == MQCC_FAILED)
{
  printf("<p>unable to open queue for output</p>\n");
}

/*****************************************************************/
/*                                                               */
/*   Read lines from the file and put them to the message queue  */
/*   Loop until null line or end of file, or there is a failure  */
/*                                                               */
/*****************************************************************/
CompCode = OpenCode;        /* use MQOPEN result for initial test */
fp = stdin;


/*****************************************************************/
/*                                                               */
/*   Put key to remote QMGR                                      */
/*                                                               */
/*****************************************************************/
md = mdDefault;

md.MsgType=MQMT_REQUEST;
memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));
md.Report = MQRO_PASS_MSG_ID + MQRO_PASS_CORREL_ID;
```

Figure 26 (Part 6 of 17). Web-Enabled MQSeries Program webmqmain

```
memcpy(md.Format,          /* character string format        */
       MQFMT_STRING, MQ_FORMAT_LENGTH);

strncpy(md.ReplyToQ, replyQ, MQ_Q_NAME_LENGTH);
strncpy(md.ReplyToQMgr, replyQMgr, MQ_Q_MGR_NAME_LENGTH);

pmo.Options=MQPMO_NO_SYNCPOINT;

strcpy(buffer, Key);
buflen=sizeof(Key);
/* to RS/60001 */
MQPUT(Hcon,                /* connection handle              */
      Hobj,                /* object handle                  */
      &md,                 /* message descriptor             */
      &pmo,                /* default options (datagram)     */
      buflen,              /* buffer length                  */
      buffer,              /* message buffer                 */
      &CompCode,           /* completion code                */
      &Reason);            /* reason code                    */


/* report reason, if any */
if (Reason != MQRC_NONE)
{
  printf("<p>MQPUT ended with reason code %ld</p>\n", Reason);
}

memcpy(saveMsgId, md.MsgId, sizeof(md.MsgId));
memcpy(saveCorrelId, md.CorrelId, sizeof(md.CorrelId));

/*********************************************************************/
/*                                                                   */
/*   Close the target queue (if it was opened)                       */
/*                                                                   */
/*********************************************************************/
```

Figure 26 (Part 7 of 17). Web-Enabled MQSeries Program webmqmain

```
   if (OpenCode != MQCC_FAILED)
   {
     C_options = 0;                  /* no close options             */
     MQCLOSE(Hcon,                   /* connection handle            */
             &Hobj,                   /* object handle               */
             C_options,
             &CompCode,               /* completion code             */
             &Reason);                /* reason code                 */

     /* report reason, if any     */
     if (Reason != MQRC_NONE)
     {
       printf("<p>MQCLOSE ended with reason code %ld</p>\n", Reason);
     }
   }

   /*-----------------------------------------------------------------*/



   /*******************************************************************/
   /*                                                                 */
   /*   Use parameter as the name of the target queue                 */
   /*                                                                 */
   /*******************************************************************/
   strcpy(od.ObjectName, "WEB.REQUESTQ.CSQ2");

/*
printf("<p>target queue is %s</p>\n", od.ObjectName);
*/


   /*******************************************************************/
   /*                                                                 */
   /*   Open the target message queue for output                      */
   /*                                                                 */
   /*******************************************************************/
```

Figure 26 (Part 8 of 17). Web-Enabled MQSeries Program webmqmain

```
O_options = MQOO_OUTPUT           /* open queue for output        */
       + MQOO_FAIL_IF_QUIESCING; /* but not if MQM stopping      */
MQOPEN(Hcon,                      /* connection handle            */
       &od,                       /* object descriptor for queue  */
       O_options,                 /* open options                 */
       &Hobj,                     /* object handle                */
       &OpenCode,                 /* MQOPEN completion code       */
       &Reason);                  /* reason code                  */

/* report reason, if any; stop if failed      */
if (Reason != MQRC_NONE)
{
  printf("<p>MQOPEN ended with reason code %ld</p>\n", Reason);
}

if (OpenCode == MQCC_FAILED)
{
  printf("<p>unable to open queue for output</p>\n");
}

/*********************************************************************/
/*                                                                   */
/*   Read lines from the file and put them to the message queue      */
/*   Loop until null line or end of file, or there is a failure      */
/*                                                                   */
/*********************************************************************/
CompCode = OpenCode;        /* use MQOPEN result for initial test */
fp = stdin;


/*********************************************************************/
/*                                                                   */
/*   Put key to remote QMGR                                          */
/*                                                                   */
/*********************************************************************/
md = mdDefault;
```

Figure 26 (Part 9 of 17). Web-Enabled MQSeries Program webmqmain

```
md.MsgType=MQMT_REQUEST;
memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));

md.Report = MQRO_PASS_MSG_ID + MQRO_PASS_CORREL_ID;


memcpy(md.Format,          /* character string format         */
       MQFMT_STRING, MQ_FORMAT_LENGTH);

strncpy(md.ReplyToQ, replyQ, MQ_Q_NAME_LENGTH);
strncpy(md.ReplyToQMgr, replyQMgr, MQ_Q_MGR_NAME_LENGTH);

pmo.Options=MQPMO_NO_SYNCPOINT;

strcpy(buffer, Key);
buflen=sizeof(Key);

/* to HOST */
MQPUT(Hcon,                /* connection handle               */
      Hobj,                /* object handle                   */
      &md,                 /* message descriptor              */
      &pmo,                /* default options (datagram)      */
      buflen,              /* buffer length                   */
      buffer,              /* message buffer                  */
      &CompCode,           /* completion code                 */
      &Reason);            /* reason code                     */


/* report reason, if any */
if (Reason != MQRC_NONE)
{
  printf("<p>MQPUT ended with reason code %ld</p>\n", Reason);
}

memcpy(saveMsgId1, md.MsgId, sizeof(md.MsgId));
memcpy(saveCorrelId1, md.CorrelId, sizeof(md.CorrelId));
```

Figure 26 (Part 10 of 17). Web-Enabled MQSeries Program webmqmain

```
/*****************************************************************/
/*                                                               */
/*   Close the target queue (if it was opened)                   */
/*                                                               */
/*****************************************************************/
if (OpenCode != MQCC_FAILED)
{
  C_options = 0;                  /* no close options            */
  MQCLOSE(Hcon,                   /* connection handle           */
          &Hobj,                   /* object handle              */
          C_options,
          &CompCode,               /* completion code            */
          &Reason);                /* reason code                */

  /* report reason, if any     */
  if (Reason != MQRC_NONE)
  {
    printf("<p>MQCLOSE ended with reason code %ld</p>\n", Reason);
  }
}


/*---------------------------------------------------------------*/


/*****************************************************************/
/*                                                               */
/*   Open the named message queue for input; exclusive or shared */
/*   use of the queue is controlled by the queue definition here */
/*                                                               */
/*****************************************************************/

strcpy(od.ObjectName, "WEB.REPLYQ");
O_options = MQOO_INPUT_AS_Q_DEF   /* open queue for input        */
      + MQOO_FAIL_IF_QUIESCING;   /* but not if MQM stopping     */
```

Figure 26 (Part 11 of 17). Web-Enabled MQSeries Program webmqmain

```
   MQOPEN(Hcon,                      /* connection handle            */
          &od,                       /* object descriptor for queue  */
          O_options,                 /* open options                 */
          &Hobj,                     /* object handle                */
          &OpenCode,                 /* completion code              */
          &Reason);                  /* reason code                  */

   /* report reason, if any; stop if failed      */
   if (Reason != MQRC_NONE)
   {
     printf("<p>MQOPEN ended with reason code %ld</p>\n", Reason);
   }

   if (OpenCode == MQCC_FAILED)
   {
     printf("<p>unable to open queue for input</p>\n");
   }

   /*****************************************************************/
   /*                                                               */
   /*   Get messages from the message queue                         */
   /*   Loop until there is a failure                               */
   /*                                                               */
   /*****************************************************************/
   CompCode = OpenCode;       /* use MQOPEN result for initial test  */

   buflen = sizeof(buffer) - 1; /* buffer size available for GET   */



/*
printf("<p>buflen=%d</p>\n", buflen);
*/
```

Figure 26 (Part 12 of 17). Web-Enabled MQSeries Program webmqmain

```
gmo.Options = MQGMO_WAIT + MQGMO_ACCEPT_TRUNCATED_MSG + MQGMO_CONVERT;
                                  /* wait for new messages           */
gmo.WaitInterval = 30000;    /* 50 second limit for waiting     */

/*****************************************************************/
/*                                                               */
/*   In order to read the messages in sequence, MsgId and        */
/*   CorrelID must have the default value.  MQGET sets them      */
/*   to the values in for message it returns, so re-initialise   */
/*   them before every call                                      */
/*                                                               */
/*****************************************************************/
memcpy(md.MsgId, saveMsgId, sizeof(md.MsgId));
memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));


/* Get from RS6000 */
MQGET(Hcon,                /* connection handle                */
      Hobj,                /* object handle                    */
      &md,                 /* message descriptor               */
      &gmo,                /* get message options              */
      buflen,              /* buffer length                    */
      buffer,              /* message buffer                   */
      &messlen,            /* message length                   */
      &CompCode,           /* completion code                  */
      &Reason);            /* reason code                      */


/* report reason, if any     */
if (Reason != MQRC_NONE)
{
  if (Reason == MQRC_NO_MSG_AVAILABLE)
  {                         /* special report for normal end    */
    printf("<p>no more messages from RS60001</p>\n");
  }
  else                      /* general report for other reasons */
  {
```

Figure 26 (Part 13 of 17). Web-Enabled MQSeries Program webmqmain

```
       printf("<p>MQGET ended with reason code %ld</p>\n", Reason);

       /*   treat truncated message as a failure for this sample   */
       if (Reason == MQRC_TRUNCATED_MSG_FAILED)
       {
         CompCode = MQCC_FAILED;
       }
     }
   }


   /*********************************************************/
   /*   Display message received                            */
   /*********************************************************/
   if (CompCode != MQCC_FAILED)
   {
     buffer[messlen] = '\0';             /* add terminator          */


/*
printf("<p>message from RS60001: %s</p>\n", buffer);
*/




     /* HTML */
     /* get image file name from message */
     i=10;
     j=0;
     do {
        filename[j]=buffer[i];
        i++;
        j++;
     } while (buffer[i] != '\0');
     filename[i-10]='\0';
   }
```

Figure 26 (Part 14 of 17). Web-Enabled MQSeries Program webmqmain

```
/*
printf("<p>The name of filename you choose is %s</p>\n", filename);
*/
   printf("<center>\n");
   printf("<img src=/images/%s>\n", filename);
   printf("<hr noshade size=1 width=545 align=center>\n");
   printf("</center>\n");

   printf("<TD>\n");
   CompCode = OpenCode;       /* use MQOPEN result for initial test  */

   buflen = sizeof(buffer) - 1;   /* buffer size available for GET   */
   gmo.Options = MQGMO_WAIT + MQGMO_ACCEPT_TRUNCATED_MSG + MQGMO_CONVERT;
                                  /* wait for new messages           */
   gmo.WaitInterval = 30000;      /* 50 second limit for waiting     */

   /*****************************************************************/
   /*                                                               */
   /*   In order to read the messages in sequence, MsgId and        */
   /*   CorrelID must have the default value.  MQGET sets them      */
   /*   to the values in for message it returns, so re-initialise   */
   /*   them before every call                                      */
   /*                                                               */
   /*****************************************************************/
   memcpy(md.MsgId, saveMsgId1, sizeof(md.MsgId));
   memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));


   /* Get from HOST */
   MQGET(Hcon,                /* connection handle                */
         Hobj,                /* object handle                    */
         &md,                 /* message descriptor               */
         &gmo,                /* get message options              */
         buflen,              /* buffer length                    */
         buffer,              /* message buffer                   */
         &messlen,            /* message length                   */
         &CompCode,           /* completion code                  */
         &Reason);            /* reason code                      */
```

Figure 26 (Part 15 of 17). Web-Enabled MQSeries Program webmqmain

```
/* report reason, if any     */
if (Reason != MQRC_NONE)
{
  if (Reason == MQRC_NO_MSG_AVAILABLE)
  {                          /* special report for normal end    */
    printf("<p>No more messages from CSQ2\n");
    printf("<p>Please check the connection to CSQ2\n");
  }
  else                       /* general report for other reasons */
  {
    printf("<p>MQGET ended with reason code %ld\n", Reason);

    /*   treat truncated message as a failure for this sample   */
    if (Reason == MQRC_TRUNCATED_MSG_FAILED)
    {
      CompCode = MQCC_FAILED;
    }
  }
}


/*********************************************************/
/*   Display message received                            */
/*********************************************************/
if (CompCode != MQCC_FAILED)
{
  buffer[messlen] = '\0';            /* add terminator          */

  /* display the description messages from CSQ2 */
  printf("<p>%s\n", buffer);

}
/*********************************************************************/
/*                                                                   */
/*   Close the source queue (if it was opened)                       */
/*                                                                   */
/*********************************************************************/
```

Figure 26 (Part 16 of 17). Web-Enabled MQSeries Program webmqmain

```
   if (OpenCode != MQCC_FAILED)
   {
     C_options = 0;                  /* no close options           */
     MQCLOSE(Hcon,                   /* connection handle          */
             &Hobj,                  /* object handle              */
             C_options,
             &CompCode,              /* completion code            */
             &Reason);               /* reason code                */

     /* report reason, if any     */
     if (Reason != MQRC_NONE)
     {
       printf("<p>MQCLOSE ended with reason code %ld\n", Reason);
     }
   }
   /*****************************************************************/
   /*                                                               */
   /*   Disconnect from MQM if not already connected                */
   /*                                                               */
   /*****************************************************************/
   if (CReason != MQRC_ALREADY_CONNECTED)
   {
     MQDISC(&Hcon,                   /* connection handle          */
            &CompCode,               /* completion code            */
            &Reason);                /* reason code                */

     /* report reason, if any     */
     if (Reason != MQRC_NONE)
     {
       printf("<p>MQDISC ended with reason code %ld</p>\n", Reason);
     }
   }
   /*****************************************************************/
   /*                                                               */
   /* END OF webmqmain                                              */
   /*                                                               */
   /*****************************************************************/
/*
printf("<p>Sample webmq end</p>\n");
*/
   printf("</TD>\n");
   printf("</BODY></HTML>\n");
   delete [] messageBuffer;
   return(0);
 }
```

Figure 26 (Part 17 of 17). Web-Enabled MQSeries Program webmqmain

A.7 Makefile

This is the makefile to compile the OS/2 program.

```
!include Makerule

qbobj= queuepro.obj queuescan.obj hqmsglist.obj hqmsgmd.obj \
        hqmsgda.obj cache.obj webmqmain.obj

 cpp.obj:
        $(CC) $(CPPFLAGS) $(CPPINC) -Ge+ $<

all: MQQueueB.exe amqwput.exe amqwget.exe MQHost.exe webmqmain.exe

amqwput.exe:amqwput0.obj
    -$(LINKER) /B"$(LFLAGS) $(EXEBIT)" $(CPPFLAGS) $(CPPINC) -C- -Ge+\
    -Fe$*.exe $** os2386.lib $(MQLIB) $(CGILIB) $(CPPINC);

amqwget.exe:   amqwget0.obj
    -$(LINKER) /B"$(LFLAGS) $(EXEBIT)" $(CPPFLAGS) $(CPPINC) -C- -Ge+\
    -Fe$*.exe $** os2386.lib $(MQLIB) $(CGILIB) $(CPPINC);

MQHost.exe:MQHost.obj queuepro.obj cache.obj
    -$(LINKER) /B"$(LFLAGS) $(EXEBIT)" $(CPPFLAGS) $(CPPINC) -C- -Ge+\
    -Fe$*.exe $** os2386.lib $(MQLIB) $(CGILIB) $(CPPINC);

MQQueueB.exe:MQQueueB.obj $(qbobj)
    -$(LINKER) /B"$(LFLAGS) $(EXEBIT)" $(CPPFLAGS) $(CPPINC) -C- -Ge+\
    -Fe$*.exe $** os2386.lib $(MQLIB) $(CGILIB) $(CPPINC);

webmqmain.exe:   webmqmain.obj
    -$(LINKER) /B"$(LFLAGS) $(EXEBIT)" $(CPPFLAGS) $(CPPINC) -C- -Ge+\
    -Fe$*.exe $** os2386.lib $(MQLIB) $(CGILIB) $(CPPINC);

clean:
 del *.obj MQQueueB.exe MQHost.exe amqwget.exe amqwput.exe webmqmain.exe
 >nul 2> 1
```

Figure 27. Makefile

A.8 Makerule

This is the makerule to compile the OS/2 program.

```
CC= icc
FFIXES:
FFIXES:.obj .cpp .h .c

#The variable MQ needs to be set to the path MQSeries is installed in
MQ=d:\mqm

LINKER=icc
LFLAGS=/DEBUG /NOE /NOL

MQINC=$(MQ)\tools\c\include
MQLIB=mqm.lib
MQCLIENT=mqic.lib
CGILIB=\mqgate\mqgate\source\cgilib\mqcgi.lib
CPPSRC= -I \mqgate\mqgate\source\gateway -I \mqgate\mqgate\source\cgilib
-I \mqgate\mqgate\source\samples
CPPINC= -I \mqgate\mqgate\source -I$(MQINC) $(CPPSRC)
EXEBIT=/STACK:65536 /PM:VIO

CPPFLAGS = -Gd- -Gm- -Gs -J- -Q -Sp1 -W1 -Gh -Ti -C+

 obj.dll:
        $(CC) -c $(CPPFLAGS) $(CPPINC) $** -Fe$*.dll

 cpp.obj:
        $(CC) -c $(CPPFLAGS) $(CPPINC) -Ge- $<
```

Figure 28. Makerule

A.9 HTMLPAGE1

This is the HTML page for the Web browser front-end.

```
<HEAD>
<TITLE>MQSeries CGI ITSO</TITLE>
</HEAD>

<BODY BGCOLOR="#E0E0FF">
<center>
<img src=/webmq/ibmlogo.gifalt="[ MQSeries CGI Demonstration ">

<hr noshade size=1 width=545 align=center>
</center>
<FORM ACTION="/cgi-bin/webmqmain" METHOD="POST">

<center><p><INPUT NAME="Key" VALUE="0000000001" maxlength=10 size=10>
</p></center>

<center><P>Please enter key to access.</P></center>

<hr noshade size=1 width=545 align=center>


<TD>
<center><A HREF="/webmq/webmqh.html">Help</A></center>
</TD>
</P>

<P>
<font size=+1>
<center><INPUT TYPE="submit" VALUE="OK"></center>
</font>
</P>
</FORM>
<hr noshade size=1 width=545 align=center>
</BODY>
</HTML>
```

Figure 29. HTMLPAGE1

A.10 HTMLPAGE2

This is the Help HTML page for the Web browser front-end.

```
<HEAD>
<TITLE>MQSeries CGI Help</TITLE>
</HEAD>

<BODY BGCOLOR="#E0E0FF">
<br>
<center>
<img src="/webmq/ibmlogo.gif">
</center>
<center>

<P><strong>MQSeries CGI Help</strong></P>

</BODY>
</HTML>
```

Figure 30. HTMLPAGE2

A.11 AIX Application

This is code for the AIX application server.

```
/*************************************************************************/
/*                                                                       */
/* Program name: webmqsrv                                                */
/*                                                                       */
/*  Program logic:                                                       */
/*     MQCONNect to message queue manager                                */
/*     MQOPEN message queue for shared input                             */
/*     while no MQI failures,                                            */
/*     .  MQGET next message from input queue                            */
/*     .  Prepare reply message if MQGET was successful                  */
/*     .  MQPUT1, send reply or report to named reply queue              */
/*     MQCLOSE queue A                                                   */
/*     MQDISConnect from queue manager                                   */
/*                                                                       */
/*                                                                       */
/*************************************************************************/
/*                                                                       */
/*   AMQSECHA has 1 parameter - a string (MQTMC2) based on the           */
/*       initiation trigger message; only the QName and queue            */
/*       manager name fields are used in this example                    */
/*                                                                       */
/*************************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
    /*  includes for MQI  */
#include <cmqc.h>

int main(int argc, char **argv)
{

  /*   Declare MQI structures needed                                    */
  MQOD    odG = {MQOD_DEFAULT};    /* Object Descriptor for GET     */
  MQOD    odR = {MQOD_DEFAULT};    /* Object Descriptor for reply   */
  MQOD    odI = {MQOD_DEFAULT};    /* Object Descriptor (INQUIRE)   */
  MQMD     md = {MQMD_DEFAULT};    /* Message Descriptor            */
  MQGMO   gmo = {MQGMO_DEFAULT};   /* get message options           */
  MQPMO   pmo = {MQPMO_DEFAULT};   /* put message options           */
     /** note, sample uses defaults where it can **/
  MQTMC2   *trig;                  /* trigger message structure     */

  MQHCONN  Hcon;                   /* connection handle             */
  MQHOBJ   Hobj;                   /* object handle, server queue   */
  MQHOBJ   Hinq;                   /* handle for MQINQ              */
  MQLONG   O_options;              /* MQOPEN options                */
```

Figure 31 (Part 1 of 8). AIX Application webmqsrv

```
MQLONG   C_options;              /* MQCLOSE options               */
MQLONG   CompCode;               /* completion code               */
MQLONG   Reason;                 /* reason code                   */
MQLONG   CReason;                /* reason code (MQCONN)          */
MQBYTE   reply[400];             /* message reply text            */
MQBYTE   buffer[100];            /* message buffer                */
MQLONG   buflen;                 /* buffer length                 */
MQLONG   messlen;                /* message length received       */
char     key[20];

printf("Sample webmqsrv start\n");
if (argc < 2)
{
  printf("Missing parameter - start program by MQI trigger\n");
  exit(99);
}

/*****************************************************************/
/*                                                               */
/*   Set the program argument into the trigger message           */
/*                                                               */
/*****************************************************************/
trig = (MQTMC2*)argv[1];        /* -> trigger message            */

/*****************************************************************/
/*                                                               */
/*   This sample includes an explicit connect (MQCONN),          */
/*   which isn't required on all MQSeries platforms,             */
/*   because it's intended to be portable                        */
/*                                                               */
/*****************************************************************/
MQCONN(trig->QMgrName,           /* queue manager                 */
       &Hcon,                    /* connection handle             */
       &CompCode,                /* completion code               */
       &CReason);                /* reason code                   */

/* report reason and stop if it failed     */
```

Figure 31 (Part 2 of 8). AIX Application webmqsrv

```
if (CompCode == MQCC_FAILED)
{
  printf("MQCONN ended with reason code %ld\n", CReason);
  exit(CReason);
}

/*****************************************************************/
/*                                                               */
/*   Open the message queue for shared input                     */
/*                                                               */
/*****************************************************************/
memcpy(odG.ObjectName,          /* name of input queue           */
       trig -> QName, MQ_Q_NAME_LENGTH);
O_options = MQOO_INPUT_SHARED   /* open queue for shared input   */
          + MQOO_FAIL_IF_QUIESCING; /* but not if MQM stopping   */
MQOPEN(Hcon,                    /* connection handle             */
       &odG,                    /* object descriptor for queue   */
       O_options,               /* open options                  */
       &Hobj,                   /* object handle                 */
       &CompCode,               /* MQOPEN completion code        */
       &Reason);                /* reason code                   */

/* report reason if any; stop if it failed     */
if (Reason != MQRC_NONE)
{
  printf("MQOPEN (input) ended with reason code %ld\n", Reason);
}

if (CompCode == MQCC_FAILED)
{
  exit(Reason);
}

/*****************************************************************/
/*                                                               */
/*   Get messages from the message queue                         */
/*   Loop until there is a warning or failure                    */
/*                                                               */
```

Figure 31 (Part 3 of 8). AIX Application webmqsrv

```
/*                                                                  */
/********************************************************************/
buflen = sizeof(buffer) - 1;
while (CompCode == MQCC_OK)
{
  gmo.Options = MQGMO_ACCEPT_TRUNCATED_MSG
              + MQGMO_CONVERT  /* receive converted messages      */
              + MQGMO_WAIT;    /* wait for new messages           */
  gmo.WaitInterval = 5000;     /* 5 second limit for waiting      */
                               /* specify representation required */
  md.Encoding = MQENC_NATIVE;
  md.CodedCharSetId = MQCCSI_Q_MGR;

  /********************************************************************/
  /*                                                                  */
  /*   In order to read the messages in sequence, MsgId and           */
  /*   CorrelID must have the default value.  MQGET sets them         */
  /*   to the values in for message it returns, so re-initialise      */
  /*   them before every call                                         */
  /*                                                                  */
  /********************************************************************/
  memcpy(md.MsgId, MQMI_NONE, sizeof(md.MsgId));
  memcpy(md.CorrelId, MQCI_NONE, sizeof(md.CorrelId));

  MQGET(Hcon,               /* connection handle                 */
        Hobj,               /* object handle                     */
        &md,                /* message descriptor                */
        &gmo,               /* GET options                       */
        buflen,             /* buffer length                     */
        buffer,             /* message buffer                    */
        &messlen,           /* message length                    */
        &CompCode,          /* completion code                   */
        &Reason);           /* reason code                       */

  /* report reason if any  (loop ends if it failed)    */
  if (Reason != MQRC_NONE)
  {
```

Figure 31 (Part 4 of 8). AIX Application webmqsrv

```
  printf("MQGET ended with reason code %ld\n", Reason);
}

/*****************************************************************/
/*                                                               */
/*    Only process REQUEST messages                              */
/*                                                               */
/*****************************************************************/
if ((CompCode == MQCC_OK)  ||  (CompCode == MQCC_WARNING))
{
  buffer[messlen] = '\0';  /* end string ready to use */
  printf("Message received:%s\n", buffer);

  if (md.MsgType != MQMT_REQUEST)
  {
    printf("  -- not a request and discarded\n");
    continue;
  }

  /*****************************************************************/
  /*                                                               */
  /*   Send reply using MQPUT1                                     */
  /*                                                               */
  /*****************************************************************/
  md.MsgType = MQMT_REPLY;

  /*****************************************************************/
  /*                                                               */
  /*   Copy the ReplyTo queue name to the object descriptor        */
  /*                                                               */
  /*****************************************************************/
  strncpy(odR.ObjectName, md.ReplyToQ, MQ_Q_NAME_LENGTH);
  strncpy(odR.ObjectQMgrName,
          md.ReplyToQMgr, MQ_Q_MGR_NAME_LENGTH);

  /*****************************************************************/
  /*                                                               */
```

Figure 31 (Part 5 of 8). AIX Application webmqsrv

```
      /*   MsgId and CorrelId are currently the values of the       */
      /*   got message.  Reset them if requested, then              */
      /*   stop further reports                                     */
      /*                                                            */
      /************************************************************/

printf("*******************************\n");
printf("received MsgId:%ld\n", md.MsgId);
printf("received CorrelId:%ld\n", md.CorrelId);
printf("received Message:*%s*\n", buffer);


      strcpy(key, buffer);
      /* get image file name for each key */
      if (!strcmp(key, "0000000001"))
         strcat(buffer,"credos.jpg");
      else
      if (!strcmp(key, "0000000002"))
         strcat(buffer,"leo.jpg");
      else
      if (!strcmp(key, "0000000003"))
         strcat(buffer,"avella2.jpg");
      else
      if (!strcmp(key, "0000000004"))
         strcat(buffer,"pr-beta.jpg");
      else
      if (!strcmp(key, "0000000005"))
         strcat(buffer,"newspo.jpg");
      else
      if (!strcmp(key, "0000000006"))
         strcat(buffer,"pregio1.jpg");
      else
      if (!strcmp(key, "0000000007"))
         strcat(buffer,"besta1.jpg");
      else
      if (!strcmp(key, "0000000008"))
         strcat(buffer,"concept1.gif");
```

Figure 31 (Part 6 of 8). AIX Application webmqsrv

```
      md.Report = MQRO_NONE;           /*    stop further reports   */
      messlen=sizeof(buffer);
printf("Message sent:%s\n", buffer);
printf("Message length sent:%d\n", messlen);
      /*****************************************************************/
      /*                                                               */
      /*   Put the message                                             */
      /*                                                               */
      /*****************************************************************/
      MQPUT1(Hcon,            /* connection handle                    */
             &odR,            /* object descriptor                    */
             &md,             /* message descriptor                   */
             &pmo,            /* default options                      */
             messlen,         /* message length                       */
             buffer,          /* message buffer                       */
             &CompCode,       /* completion code                      */
             &Reason);        /* reason code                          */

      /* report reason if any  (loop ends if it failed)    */
      if (Reason != MQRC_NONE)
      {
        printf("MQPUT1 ended with reason code %ld\n", Reason);
      }
    }     /* end message for reply               */
  }     /* end Get message loop                */


printf("sent MsgId:%ld\n", md.MsgId);
printf("sent CorrelId:%ld\n", md.CorrelId);


  /*****************************************************************/
  /*                                                               */
  /*   Close server queue when no messages left                    */
  /*                                                               */
  /*****************************************************************/
  C_options = 0;                  /* no close options             */
```

Figure 31 (Part 7 of 8). AIX Application webmqsrv

```
   MQCLOSE(Hcon,                    /* connection handle            */
           &Hobj,                   /* object handle                */
           C_options,
           &CompCode,               /* completion code              */
           &Reason);                /* reason code                  */

   /* report reason, if any     */
   if (Reason != MQRC_NONE)
   {
     printf("MQCLOSE ended with reason code %ld\n", Reason);
   }

   /*****************************************************************/
   /*                                                               */
   /*   Disconnect from MQM  (unless previously connected)          */
   /*                                                               */
   /*****************************************************************/
   if (CReason != MQRC_ALREADY_CONNECTED)
   {
     MQDISC(&Hcon,                   /* connection handle            */
            &CompCode,               /* completion code              */
            &Reason);                /* reason code                  */

     /* report reason, if any     */
     if (Reason != MQRC_NONE)
     {
       printf("MQDISC ended with reason code %ld\n", Reason);
     }
   }

   /*****************************************************************/
   /*                                                               */
   /*                                                               */
   /*****************************************************************/
   printf("Sample webmqsrv end\n");
   return(0);
}
```

Figure 31 (Part 8 of 8). AIX Application webmqsrv

Appendix B. MQSeries Definitions

This appendix contains MQSeries definitions used in the user CGI example.

B.1 MQSeries OS/2 Application Objects

QLOCAL and QREMOTE definitions for the Web server are shown below.

```
DEFINE QLOCAL('WEB.REPLYQ') REPLACE +
       DESCR('Local queue for web interface') +
       DEFPSIST(YES)  +
       SHARE
DEFINE QREMOTE('WEB.REQUESTQ.RS60001') REPLACE  +
       DESCR('Remote queue for web interface') +
       DEFPSIST(YES)                +
       RNAME('WEB.REQUESTQ.RS60001') +
       RQMNAME('RS60001')
DEFINE QREMOTE('WEB.REQUESTQ.CSQ2') REPLACE  +
       DESCR('Remote queue for web interface') +
       DEFPSIST(YES)                +
       RNAME('WEB.REQUESTQ.CSQ2') +
       RQMNAME('CSQ2')
```

Figure 32. Define OS/2 MQSeries Queue Objects

B.2 MQSeries OS/2 Connectivity Objects

Channel definitions for the Web server are shown below.

```
DEFINE CHANNEL(MQOS3.TO.RS60001)
   CHLTYPE(SDR)
   TRPTYPE(TCP)
   DESCR( )
   XMITQ(RS60001)
   MCANAME()
   MODENAME()
   TPNAME()
   BATCHSZ(50)
   DISCINT(0)
   SHORTRTY(10)
   SHORTTMR(60)
   LONGRTY(999999999)
   LONGTMR(1200)
   SCYEXIT()
   MSGEXIT()
   SENDEXIT()
   RCVEXIT()
   SEQWRAP(999999)
   MAXMSGL(4194304)
   CONVERT(NO)
   SCYDATA()
   MSGDATA()
   SENDDATA()
   RCVDATA()
   USERID()
   PASSWORD()
   MCAUSER()
   MCATYPE(PROCESS)
   CONNAME(RS60001)
```

Figure 33 (Part 1 of 3). Define OS/2 MQSeries Channel Objects

```
DEFINE CHANNEL(RS60001.TO.MQOS3)
   CHLTYPE(RCVR)
   TRPTYPE(TCP)
   DESCR( )
   BATCHSZ(50)
   SCYEXIT()
   MSGEXIT()
   SENDEXIT()
   RCVEXIT()
   SEQWRAP(999999)
   MAXMSGL(4194304)
   PUTAUT(DEF)
   SCYDATA()
   MSGDATA()
   SENDDATA()
   RCVDATA()
   MCAUSER()

DEFINE   CHANNEL(MQOS3.TO.CSQ2)
   CHLTYPE(SDR)
   TRPTYPE(LU62)
   DESCR( )
   XMITQ(CSQ2)
   MCANAME()
   MODENAME(LU62APPC)
   TPNAME(CKRC)
   BATCHSZ(50)
   DISCINT(6000)
   SHORTRTY(10)
   SHORTTMR(60)
   LONGRTY(999999999)
   LONGTMR(1200)
   SCYEXIT()
   MSGEXIT()
   SENDEXIT()
   RCVEXIT()
   SEQWRAP(999999)
   MAXMSGL(4194304)
   CONVERT(YES)
   SCYDATA()
   MSGDATA()
   SENDDATA()
   RCVDATA()
   USERID()
   PASSWORD()
   MCAUSER()
   MCATYPE(PROCESS)
   CONNAME(USIBMRA.RAIAC)
```

Figure 33 (Part 2 of 3). Define OS/2 MQSeries Channel Objects

```
DEFINE CHANNEL(CSQ2.TO.MQOS3)
   CHLTYPE(RCVR)
   TRPTYPE(LU62)
   DESCR( )
   BATCHSZ(50)
   SCYEXIT()
   MSGEXIT()
   SENDEXIT()
   RCVEXIT()
   SEQWRAP(999999)
   MAXMSGL(4194304)
   PUTAUT(DEF)
   SCYDATA()
   MSGDATA()
   SENDDATA()
   RCVDATA()
   MCAUSER()
```

Figure 33 (Part 3 of 3). Define OS/2 MQSeries Channel Objects

B.3 MQSeries AIX Application Objects

MQSeries definitions for the AIX application server are shown below.

```
DEFINE QLOCAL('WEB.REQUESTQ.RS60001') REPLACE +
   DESCR('Local queue for web interface') +
   DEFPSIST(YES)  +
   SHARE
DEFINE QLOCAL('WEB1.INITQ')  REPLACE
DEFINE PROCESS(WEB1) replace +
   APPLTYPE(AIX) +
   APPLICID('/webmq/webmqsrv')
DEFINE CHANNEL(RS60001.TO.MQOS3)
  CHLTYPE(SDR)
  TRPTYPE(TCP)
  DESCR( )
  XMITQ(MQOS3)
  MCANAME()
  MODENAME()
  TPNAME()
  BATCHSZ(50)
  DISCINT(60)
  SHORTRTY(10)
  SHORTTMR(60)
  LONGRTY(999999999)
  LONGTMR(1200)
  SCYEXIT()
  MSGEXIT()
  SENDEXIT()
  RCVEXIT()
  SEQWRAP(999999)
  MAXMSGL(4194304)
  CONVERT(NO)
  SCYDATA()
  MSGDATA()
  SENDDATA()
  RCVDATA()
  USERID()
  PASSWORD()
  MCAUSER()
  MCATYPE(PROCESS)
  CONNAME(MQOS3)
```

Figure 34 (Part 1 of 2). Define AIX MQSeries Queue Objects

```
DEFINE CHANNEL(MQOS3.TO.RS60001)
   CHLTYPE(RCVR)
   TRPTYPE(TCP)
   DESCR( )
   BATCHSZ(50)
   SCYEXIT()
   MSGEXIT()
   SENDEXIT()
   RCVEXIT()
   SENDDATA()
   RCVDATA()
   USERID()
   PASSWORD()
   MCAUSER()
   MCATYPE(PROCESS)
```

Figure 34 (Part 2 of 2). Define AIX MQSeries Queue Objects

B.4 MQSeries MVS Application Objects

QLOCAL and PROCESS definitions for the S/390 CICS application server are below.

```
DEFINE QLOCAL('WEB.REQUESTQ.CSQ2') REPLACE +
    DESCR('Local queue for web interface') +
    DEFPSIST(YES)  +
    SHARE  +
    TRIGGER +
    TRIGTYPE FIRST +
    PROCESS(WEB1) +
    INITQ('CICS.INITQ')
DEFINE PROCESS(WEB1) replace +
    APPLTYPE(CICS) +
    APPLICID('WEB1')
```

Figure 35. Define MVS MQSeries Application Objects

B.5 MQSeries MVS CICS Channel Definitions

The following are the S/390 CICS channel definition screens using the CKMC transaction.

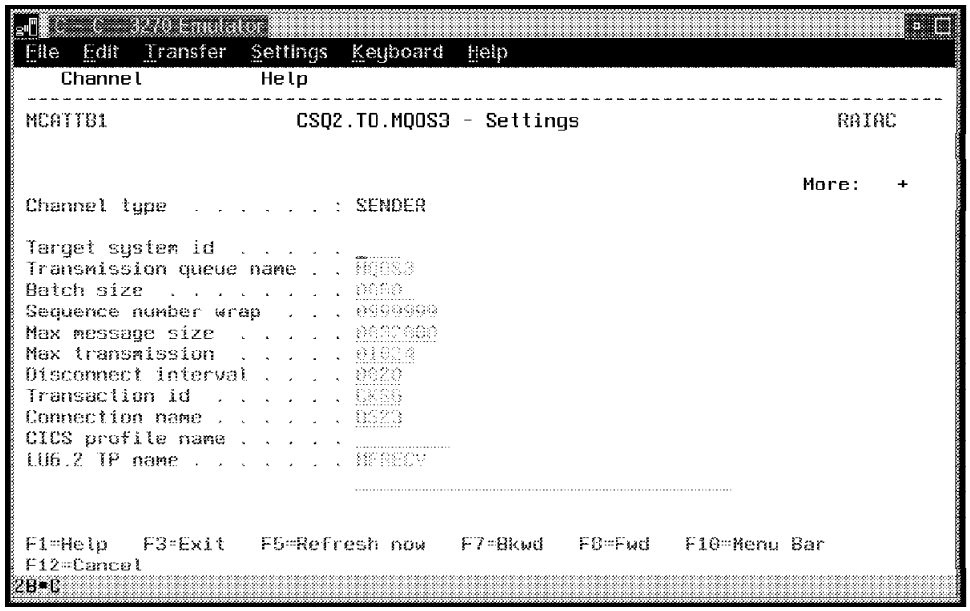


Figure 36 (Part 1 of 4). CKMC Screens for Channel Definitions

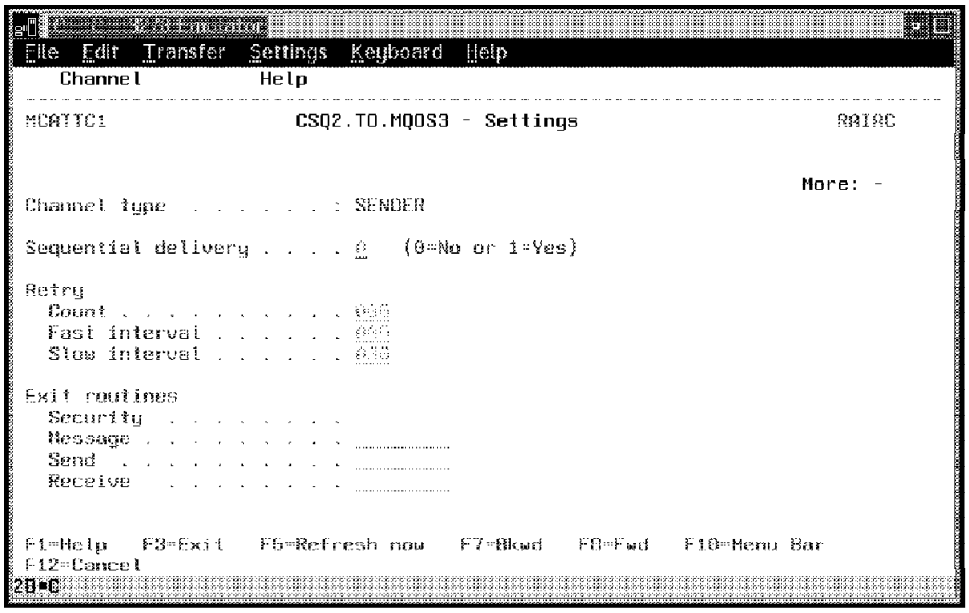


Figure 36 (Part 2 of 4). CKMC Screens for Channel Definitions

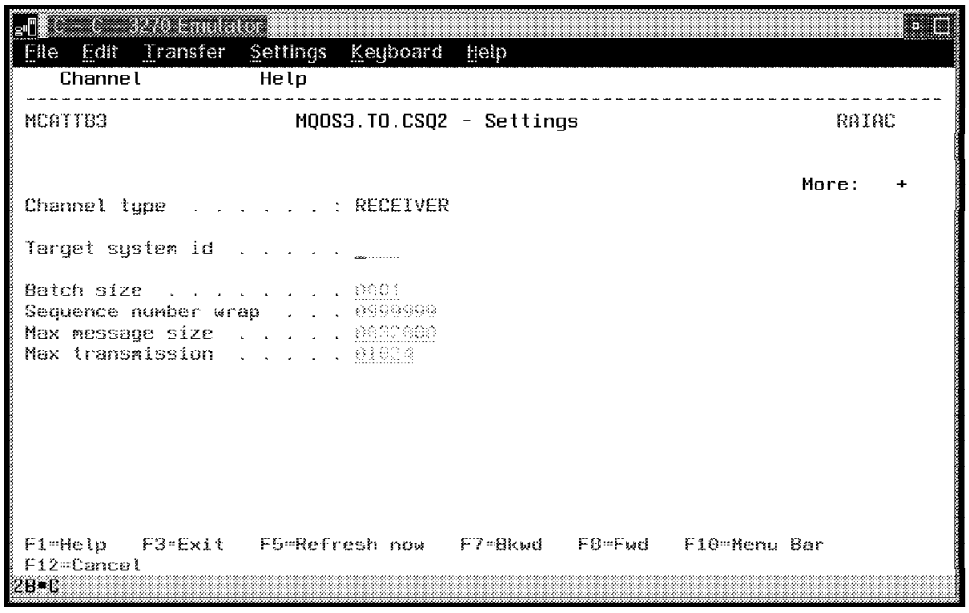


Figure 36 (Part 3 of 4). CKMC Screens for Channel Definitions

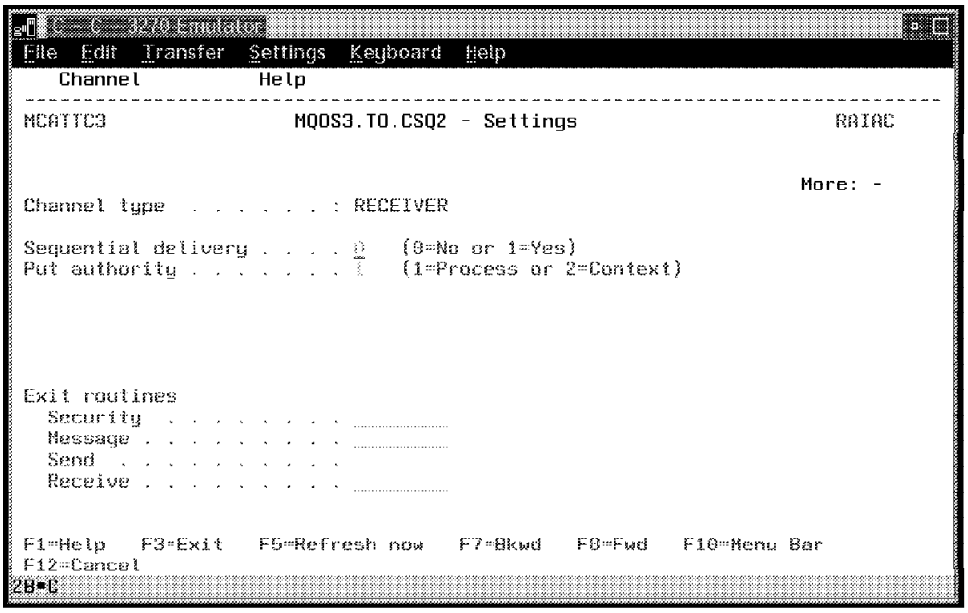


Figure 36 (Part 4 of 4). CKMC Screens for Channel Definitions

Appendix C. Source Listing and Compile Procedures

This appendix contains source listings and procedures for MVS and OS/2 used with the WTPING/WTPONG example.

C.1 HTML for the Web Front-End to WTPING/WTPONG

```
<HEAD>
<TITLE>MQSeries Internet Gateway WTPING Example</TITLE>
</HEAD>
<BODY BGCOLOR="#E0E0FF">
<center>
<A HREF="./MQGate.html">
<img src="./images/MQGate.gif" height=124 width=435
alt="[ MQGate Page ]" border=0>
</A>
<hr noshade size=1 width=545 align=center>
</center>
<FORM ACTION="/cgi-bin/MQGate" METHOD="POST">
<P>This is a sample to drive the WTPING via MQSeries Internet Gateway.
You will need a FORM capable browser.</P>
<INPUT NAME="MQIGwQueueManager" TYPE="hidden" VALUE="">
<INPUT NAME="MQIGwQueue" TYPE="hidden" VALUE="WEB.CLIENT.QUEUE">
<P>Enter the appropriate attributes and click on "SUBMIT" to
send messages on the queue.</P>
<P>No. of Msgs: <INPUT NAME="nomsg" SIZE="5" MAXLENGTH="5" VALUE="00100">
</P>
<P>Len of Msgs: <INPUT NAME="lenmsg" SIZE="6" MAXLENGTH="6" VALUE="000100">
</P>
<P>First 10Bytes:<INPUT NAME="fbytes" SIZE="10" MAXLENGTH="10"
VALUE="123456789"
</P>
<P>RemQmgr: <INPUT NAME="remqmgr" SIZE="20" MAXLENGTH="20" VALUE="CSQ2">
</P>
<P>Comment: <INPUT NAME="comment" SIZE="50" MAXLENGTH="50" VALUE="COMMENT">
</P>
<INPUT NAME="respmsg" TYPE="hidden" VALUE="00100">
<INPUT NAME="term" TYPE="hidden" VALUE="&">
<hr noshade size=1 width=545 align=center>
<P>
<font size=+1>
<INPUT TYPE="submit" VALUE="SUBMIT">
</font>
</P>
</FORM>
<hr noshade size=1 width=545 align=center>
</BODY>
</HTML>
```

Figure 37. HTML MQPING

C.2 MVS WTPING Cobol Source

```
CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
      * ------------------------------------------------------------- *
       IDENTIFICATION DIVISION.
      * ------------------------------------------------------------- *
       PROGRAM-ID. PRGQJIW1.
      *REMARKS
      * ------------------------------------------------------------- *
      * ------------------------------------------------------------- *
      * BATCH DRIVEN MQSERIES CLIENT PROGRAM------------------------ *
      *
      * THIS PROGRAM TAKES DATA FROM A PARM CARD AS WELL
      * FROM A MSGQ WEB.CLIENT.QUEUE
      * IT IS AN INTERMEDIATE CLIENT TO THE PONG-BACKEND
      * FROM THE WEB BROWSER
      *
      *
      *
      *  THIS PROGRAM USES A TEMPORARY DYNAMIC QUEUE
      *  FOR THE REPLY TO Q
      *
      *  QMODEL USED: WTPONG.REPLY.QUEUE
      *
      * ------------------------------------------------------------- *
      * ------------------------------------------------------------- *
      * CLIENT PROGRAM FOR SUBAREA SA25/18
      * NAME OF THE PROGRAM: PRGQJ I W   1:
      *                      !     ! !   !
      *                      !     ! !   +---> TEAM NAME (NOW FIXED)
      *                      !     ! +-------> CLIENT PROGRAM
      *                      !     !
      *                      !     +---------> S.A. 25
      *                      +---------------> FIXED IDENTIFIER
      *
      * PARM   (REQ  )   QUEUE: WEB.CLIENT.QUEUE(FIXED)
      * INPUT  (REPLY)   QUEUE: DXB.CSQX.REPLYQ (FIXED)
      * OUTPUT (REQUEST) QUEUE: DXB.X.REQUESTQ
      *                          !
      *                          +--------> TARGET Q MANAGER NAME
      *                                     FROM USER INPUT
      *
      * TO COPE WITH NEW QUEUE MANAGER NAMES, FIND OUT THE STRING
      * ADD HERE NEW QUEUE MANAGER NAMES
```

Figure 38 (Part 1 of 40). MVS WTPING Cobol Program Source

```
      * AND UPDATE ACCORDINGLY
      * THE STRING OCCURS TWICE
      * ------------------------------------------------------------ *
      * ----HOW THE INPUT PARMS ARE SPECIFIED ---------------------- *
      * ------------------------------------------------------------ *
      *  MESSAGE INCLUDES AN ARRAY OF THE FOLLOWING CARDS
      *  1.CARD  NO OF MESSAGES TO SEND (5 BYTES)
      *  2.CARD  MESSAGE TEXT FIRST 10 BYTES (10 BYTES)
      *  3.CARD  LENGTH OF MESSAGE  (6 BYTES) 10,100,1000,10000
      *  4.CARD  RESPOND AFTER NO.OF MESSAGES (5 BYTES)
      *  5.CARD  COMMENT (50 BYTES)
      *  6.CARD  TGTQMGR (48 BYTES)
      *                    PROGRAM LOGIC
      *                   ---------------
      *START.
      *    GET WEB.CLIENT.QUEUE GET PARAMETERS
      *
      *    GET CURRENT DAY (FOR DISPLAY)
      *
      *               PERFORM VALIDATE-INPUT
      *               IF NOT INPUT-OK
      *                  WRITE ERROR MESSAGE TO SCREEN
      *               ELSE
      *                  PERFORM OPEN QUEUES (EXIT IF ERR.)
      *                  PERFORM PROCESS-REQUEST
      *                  IF ERROR OCCURS
      *                     WRITE ERROR MESSAGE TO SCREEN
      *                  ELSE
      *                     WRITE COMPLETION MESSAGE TO SCREEN
      *                     PERFORM CLOSE QUEUES
      *
      *          WHEN OTHER (OTHER THAN PF3 OR ENTER KEY)
      *               DISPLAY ERROR MESSAGE TO SCREEN
      *
      *       END-EVALUATE
      *
      *       DISPLAY THE SCREEN MAP AND WAIT FOR INPUT DATA
      *    END-PERFORM
      *
      *    CLOSE ALL QUEUES
```

Figure 38 (Part 2 of 40). MVS WTPING Cobol Program Source

```
      *
      *
      *PROCESS-REQUEST.
      *
      *    BUILD MESSAGE
      *    PERFORM UNTIL ALL MESSAGES PROCESSED OR MQI CALL FAILED
      *        IF 1ST MSG
      *           MSGID    = FIRST
      *           CORRELID = CURRENT-DATE / TIME
      *        ELSE IF NOT-FIRST MSG
      *                MSGID    = NOTFIRST
      *                CORRELID = CORRELID OF 1ST MSG
      *             ELSE (LST MSG)
      *                MSGID    = LAST
      *                CORRELID = CORRELID OF 1ST MSG
      *             END-IF
      *        END-IF
      *
      *        PUT MESSAGE TO REQUEST QUEUE
      *        IF 1ST MSG
      *           TAKE TIMESTAMP AND ACKNOWLEDGE THE USER
      *           PERFORM  GET-RESPQ
      *        ELSE
      *           IF NOT-1ST MSG
      *              ACKNOWLEDGE THE USER FOR EVERY N MESSAGE (USER-DEF)
      *           ELSE (LAST MSG)
      *              TAKE TIMESTAMP AND ACKNOWLEDGE THE USER
      *              PERFORM  GET-RESPQ
      *           END-IF
      *        END-IF
      *    END-PERFORM
      *
      *    PERFORM GET-RESPQ UNTIL BOTH 1ST & LAST MSG GOT RESPONSE MSG
      *    IF NO CORRESPING MSG FOUND IN RESPONSE Q (LOOP FOR 500TIMES)
      *    IF NO CORRESPING MSG AGAIN, WAIT 1 SECS AND LOOP AGAIN
      *    IF STILL NO RESPONSE, WRITE ERROR MESSAGE TO SCREEN
      *
      *GET-RESPQ.
      *    GET CORRESPONDING MESSAGE (DEP. OM MSGID & CORRELID) FROM
      *        RESPONSE QUEUE
```

Figure 38 (Part 3 of 40). MVS WTPING Cobol Program Source

```
      *
      *    TAKE TIMESTAMP AND ACKNOWLEDGE THE USER
      *
      *(NOTE: THE PUTTIME IN MQMD IS IN GMT FORMAT, NEED TO CONVERT TO
      *       CURRENT TIME, THE TIME DIFFERENCE IS 5 HR AHEAD (EST.)
      *                                         IS 4 HR AHEAD (EDT.)
      * ------------------------------------------------------------ *
       ENVIRONMENT DIVISION.
       DATA        DIVISION.
      * ------------------------------------------------------------ *
      * ------------------------------------------------------------ *
       WORKING-STORAGE SECTION.
      * ------------------------------------------------------------ *
      *
      *    WOO - GENERAL WORK FIELDS
      *
       01  OUTMSG                      PIC X(80) VALUE SPACE.
       01  WOO-PARM-BUFFER.
           05  WOO-PARM-HDR     PIC X(47).
           05  WOO-PARM-STRING  PIC X(250).
      *
      *    DYNAMIC ARRAY
      *
       01  WOO-PARM-ARRAY.
           05  WOO-PARM-TABLE OCCURS   50 TO 300
               DEPENDING ON WO3-DATALEN-VALID.
               10 FILLER PIC X(1).
       01  WOO-WORK-FIELD.
           03  WOO-MESSAGE             PIC X(78)  VALUE  SPACES.
           03  WOO-MESSAGE-OUT         PIC X(80)  VALUE  SPACES.
           03  WOO-CNT-1               PIC 9(7)   VALUE  0.
           03  WOO-LOOP-CNT            PIC 9(5)   VALUE  0.
           03  WOO-COMMENT             PIC X(65) VALUE SPACE.
           03  WOO-FIRSTTXT            PIC X(10) VALUE SPACE.
           03  WOO-MSG-CNT             PIC 9(5)   VALUE  0.
           03  WOO-MSG-SEQ             PIC 9(5)   VALUE  0.
           03  WOO-MSG-PROC            PIC 9(5)   VALUE  0.
           03  WOO-MSG-NO-PROC         PIC 9(5)   VALUE  0.
           03  WOO-FST-CORRELID        PIC X(24)  VALUE  SPACES.
           03  WOO-FST-MSGID           PIC X(8)   VALUE  SPACES.
```

Figure 38 (Part 4 of 40). MVS WTPING Cobol Program Source

```
           03  WOO-LST-CORRELID        PIC X(24)  VALUE  SPACES.
           03  WOO-LST-MSGID           PIC X(8)   VALUE  SPACES.
       *
           03  WOO-MQMD-PUTDATE.
               05  WOO-PUT-YYYY        PIC X(4)   VALUE  SPACES.
               05  WOO-PUT-MM          PIC XX     VALUE  SPACES.
               05  WOO-PUT-DD          PIC XX     VALUE  SPACES.
       *
           03  WOO-MQMD-PUTTIME.
               05  WOO-PUT-HR          PIC XX     VALUE  SPACES.
               05  WOO-PUT-HR-NUM REDEFINES WOO-PUT-HR   PIC 99.
               05  WOO-PUT-MIN         PIC XX     VALUE  SPACES.
               05  WOO-PUT-SEC         PIC XX     VALUE  SPACES.
               05  WOO-PUT-TH          PIC XX     VALUE  SPACES.
       *
           03  WOO-GMT-DIFF            PIC 99     VALUE  4.
       *
           03  WOO-CURRENT-DATE.
               05  WOO-CUR-YYYY        PIC 9(04).
               05  WOO-CUR-MM          PIC XX.
               05  WOO-CUR-DD          PIC XX.
       *
           03  WOO-TEMP-PUTDATE.
               05  WOO-TEMP-PUT-DD     PIC XX     VALUE  SPACES.
               05                      PIC X      VALUE  '-'.
               05  WOO-TEMP-PUT-MM     PIC XX     VALUE  SPACES.
               05                      PIC X      VALUE  '-'.
               05  WOO-TEMP-PUT-YYYY   PIC X(4)   VALUE  SPACES.
       *
           03  WOO-TEMP-PUTTIME.
               05  WOO-TEMP-PUT-HR     PIC XX     VALUE  SPACES.
               05                      PIC X      VALUE  ':'.
               05  WOO-TEMP-PUT-MIN    PIC XX     VALUE  SPACES.
               05                      PIC X      VALUE  ':'.
               05  WOO-TEMP-PUT-SEC    PIC XX     VALUE  SPACES.
               05                      PIC X      VALUE  '.'.
               05  WOO-TEMP-PUT-TH     PIC XX     VALUE  SPACES.
       *
            01  WO5-DATESEP             PIC X VALUE '-'.
            01  WO5-TIMESEP             PIC X VALUE ':'.
```

Figure 38 (Part 5 of 40). MVS WTPING Cobol Program Source

```
       01  WO5-YEAR                PIC X(4).
       01  WO5-DATETIME.
           03  WO5-DATE.
              05 WO5-YY             PIC X(2).
              05 WO5-MM             PIC X(2).
              05 WO5-DD             PIC X(2).
              05 FILLER             PIC X(2).
           03  WO5-TIME.
              05  WO5-HH               PIC X(2).
              05  WO5-MI               PIC X(2).
              05  WO5-SS               PIC X(2).
              05  WO5-TT               PIC X(2).
       01  WOO-DATE-TIME.
           03  WOO-ABS-TIME            PIC S9(15) VALUE +0.
           03  WOO-TEMP-DATETIME.
               05  WOO-TEMP-DATE.
                   10  WOO-TEMP-DD     PIC XX     VALUE SPACES.
                   10                   PIC X      VALUE SPACES.
                   10  WOO-TEMP-MM     PIC XX     VALUE SPACES.
                   10                   PIC XXX    VALUE SPACES.
               05  WOO-TEMP-TIME       PIC X(8)   VALUE SPACES.
           03  WOO-TEMP-YEAR           PIC 9(5)   COMP  VALUE 0.
      *
       01  WOO-NUMERIC-FIELDS.
           03  WOO-MSG-LEN             PIC X(6)   VALUE  SPACES.
           03              REDEFINES WOO-MSG-LEN.
               05  WOO-MSG-LENGTH      PIC 9(6).
           03  WOO-MSG-TOTAL           PIC X(5)   VALUE  SPACES.
           03              REDEFINES WOO-MSG-TOTAL.
               05  WOO-MSG-TOT         PIC 9(5).
           03  WOO-MSG-GETRSP          PIC X(5)   VALUE  SPACES.
           03              REDEFINES WOO-MSG-GETRSP.
               05  WOO-GETRSP          PIC 9(5).
      *
       01  WOO-SCREEN-FORMAT2.
           03  FSTREQ.
               05  FILLER              PIC X(01)  VALUE SPACES.
               05  FILLER              PIC X(21)
                   VALUE 'FIRST REQUEST:  PUT  '.
               05  WOO-FSTREQ-PUTTIME  PIC X(11) VALUE SPACES.
```

Figure 38 (Part 6 of 40). MVS WTPING Cobol Program Source

```
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-FSTREQ-PUTDATE  PIC X(10) VALUE SPACES.
           05                      PIC X(03) VALUE SPACES.
           05  WOO-FSTREQ-GETTEXT  PIC X(06) VALUE SPACES.
           05  WOO-FSTREQ-GETTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-FSTREQ-GETDATE.
               10                  PIC X(6)  VALUE SPACES.
               10  WOO-FSTREQ-GYR  PIC X(4)  VALUE SPACES.
       03  FSTREP.
           05  FILLER              PIC X(01)  VALUE SPACES.
           05  FILLER              PIC X(21)
               VALUE 'FIRST REPLY  :  GET  '.
           05  WOO-FSTREP-GETTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-FSTREP-GETDATE.
               10                  PIC X(6)  VALUE SPACES.
               10  WOO-FSTREP-GYR  PIC X(4)  VALUE SPACES.
           05  FILLER              PIC X(09) VALUE '    PUT  '.
           05  WOO-FSTREP-PUTTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-FSTREP-PUTDATE  PIC X(10) VALUE SPACES.
       03  LSTREQ.
           05  FILLER              PIC X(01)  VALUE SPACES.
           05  FILLER              PIC X(21)
               VALUE 'LAST  REQUEST:  PUT  '.
           05  WOO-LSTREQ-PUTTIME  PIC X(11) VALUE SPACES.
           05                      PIC X(02) VALUE SPACES.
           05  WOO-LSTREQ-PUTDATE  PIC X(10) VALUE SPACES.
           05                      PIC X(03) VALUE SPACES.
           05  WOO-LSTREQ-GETTEXT  PIC X(06) VALUE SPACES.
           05  WOO-LSTREQ-GETTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-LSTREQ-GETDATE.
               10                  PIC X(6)  VALUE SPACES.
               10  WOO-LSTREQ-GYR  PIC X(4)  VALUE SPACES.
       03  LSTREP.
           05  FILLER              PIC X(01)  VALUE SPACES.
           05  FILLER              PIC X(21)
               VALUE 'LAST  REPLY  :  GET  '.
```

Figure 38 (Part 7 of 40). MVS WTPING Cobol Program Source

```
           05  WOO-LSTREP-GETTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-LSTREP-GETDATE.
               10                  PIC X(6)  VALUE SPACES.
               10  WOO-LSTREP-GYR  PIC X(4)  VALUE SPACES.
           05  FILLER              PIC X(09) VALUE '    PUT  '.
           05  WOO-LSTREP-PUTTIME  PIC X(11) VALUE SPACES.
           05  FILLER              PIC X(02) VALUE SPACES.
           05  WOO-LSTREP-PUTDATE  PIC X(10) VALUE SPACES.
       03  MSG-PROC.
           05  FILLER              PIC X(1)  VALUE SPACE.
           05  NO-PROC             PIC X(6)  VALUE SPACE.
           05  FILLER              PIC X(4)  VALUE ' OF '.
           05  TOT-MSG             PIC X(6)  VALUE SPACES.
           05  FILLER              PIC X(21)
               VALUE ' MESSAGES PROCESSED'.
           05  FILLER              PIC X(2)  VALUE SPACES.
           05  FILLER              PIC X(17)
               VALUE ' CLIENT PUT UPD. '.
           05  KD-TIST             PIC X(14) VALUE SPACES.
           05  KD-QMGR             PIC X(08) VALUE SPACES.
      *
      *    WO2 - QUEUE NAME FIELD
      *
       01  WO2-QUEUE-NAME.
           03  WO2-REQ-Q           PIC X(48)
           VALUE 'DUMMY                                          '.
           03  WO2-RESP-Q          PIC X(48)
           VALUE '                                               '.
           03  WO2-MODEL-Q         PIC X(48)
           VALUE 'WTPONG.REPLY.QUEUE                             '.
       01  WO2-MQM                 PIC X(48) VALUE SPACE.
       01  WO2-CLIENT-QNAME        PIC X(48) VALUE SPACE.
       01  EOF-FLAG                PIC X(01) VALUE SPACE.
       01  REC-COUNT               PIC S9(4) BINARY VALUE ZERO.
      *
      *    QUEUE MANAGER NAMES
      * ADD HERE NEW QUEUE MANAGER NAMES
      *
       01  WO2-QMGR-NAME.
```

Figure 38 (Part 8 of 40). MVS WTPING Cobol Program Source

```
           03  QMGR-CSQ1               PIC X(48)
           VALUE 'CSQ1                                    '.
           03  QMGR-CSQ2               PIC X(48)
           VALUE 'CSQ2                                    '.
           03  QMGR-MQOS1              PIC X(48)
           VALUE 'MQOS1                                   '.
           03  QMGR-MQOS2              PIC X(48)
           VALUE 'MQOS2                                   '.
           03  QMGR-MQOS3              PIC X(48)
           VALUE 'MQOS3                                   '.
           03  QMGR-RS60001            PIC X(48)
           VALUE 'RS60001                                 '.
           03  QMGR-RS60002            PIC X(48)
           VALUE 'RS60002                                 '.
           03  QMGR-RS60003            PIC X(48)
           VALUE 'RS60003                                 '.
           03  QMGR-RS600010           PIC X(48)
           VALUE 'RS600010                                '.
           03  QMGR-DEFAULT            PIC X(48)
           VALUE 'DEFAULT                                 '.
       *
       *    WO3 - MQM API FIELD
       *
        01  WO3-API-FILED.
           03  WO3-MAX-REQUESTS        PIC S9(9)  BINARY VALUE 10.
           03  WO3-REQUESTS            PIC S9(9)  BINARY VALUE 0.
           03  WO3-COMPCODE            PIC S9(9)  BINARY VALUE 0.
           03  WO3-REASON              PIC S9(9)  BINARY VALUE 0.
           03  WO3-OPTIONS             PIC S9(9)  BINARY.
           03  WO3-HCONN               PIC S9(9)  BINARY.
           03  WO3-HOBJ-CLIENTQ        PIC S9(9)  BINARY.
           03  WO3-HOBJ-CLIENT-RESPQ   PIC S9(9)  BINARY.
           03  WO3-CLIENT-CORRELID     PIC X(24)  VALUE  SPACES.
           03  WO3-CLIENT-MSGID        PIC X(24)  VALUE  SPACES.
           03  WO3-CLIENT-RESP-QNAME   PIC X(48)  VALUE  SPACES.
           03  WO3-CLIENT-RESP-QMGR    PIC X(48)  VALUE  SPACES.
           03  WO3-HOBJ-REQUESTQ       PIC S9(9)  BINARY.
           03  WO3-HOBJ-RESPQ          PIC S9(9)  BINARY.
           03  WO3-DATALEN             PIC S9(9)  BINARY.
           03  WO3-DATALEN-VALID       PIC S9(9)  BINARY.
           03  WO3-BUFFLEN             PIC S9(9)  BINARY.
           03  WO3-SELECTORCOUNT       PIC S9(9)  BINARY VALUE 1.
           03  WO3-INTATTRCOUNT        PIC S9(9)  BINARY VALUE 1.
           03  WO3-CHARATTRLENGTH      PIC S9(9)  BINARY VALUE 0.
```

Figure 38 (Part 9 of 40). MVS WTPING Cobol Program Source

```
           03  WO3-CHARATTRS           PIC X(48)  VALUE  LOW-VALUES.
           03  WO3-SELECTOR-TABLE.
               05  WO3-SELECTORS       PIC S9(9)  BINARY OCCURS 2.
           03  WO3-INTATTRS-TABLE.
               05  WO3-INTATTRS        PIC S9(9)  BINARY OCCURS 2.
      *
      * Some of the following data would not show correctly when
      * this redbook is viewed online or via CD.  The field has
      * ASCII representation, their hex values being:
      *
      *  ASCII
      *  text   C o n t e n t - T y p e :     t e x t / p l a i n
      *  (28 characters of 33 defined in source code program)
      *   Hex   4 6 6 7 6 6 7 2 5 7 7 6 3 2 2 7 6 7 7 2 7 6 6 6 0 0 0
      *  Value  3 F E 4 5 E 4 D 4 9 0 5 A 0 0 4 5 8 4 F 0 C 1 9 E A A A
      *
      *  (characters of 29 - 33 from the source code program)
      *   Hex   7 6 7 7 2
      *  Value  4 5 8 4 0
      *
      *  ASCII  t e x t
      *  text
      *
       01  WO3-REPLY-CLIENT-MSG.
           03  WO3-REPLY-HTML          PIC X(34) VALUE
      * Content-Type:  text/plain   text
                '***  See Above Comment   ********'.
           03  WO3-REPLY-CLIENT-MSG2.
               05  WO3-REPLY-FSTREP        PIC X(100) VALUE SPACE.
               05  WO3-REPLY-LSTREP        PIC X(100) VALUE SPACE.
               05  WO3-REPLY-NOPROC        PIC X(100) VALUE SPACE.
           03  FILLER                  PIC X(1) VALUE ''.
       01  WO3-RESP-HDR.
           03  WO3-RESP-MSGID          PIC X(24).
               88 WO3-RESP-FIRST-MSG
               VALUE 'FIRST                  '.
               88 WO3-RESP-INTMED-MSG  VALUE 'NOTFIRST'.
               88 WO3-RESP-LAST-MSG
               VALUE 'LAST                   '.
           03  WO3-RESP-CORRELID       PIC X(24).
      *
       01  WO3-GET-BUFFER.
           COPY PRGQO1X2.
      **   03  WO3-RESPONSE.
      **       05  WO3-RSP-REQ-GETTIME.
      **           10  WO3-PSP-REQ-GETDATE  PIC X(8).
      **           10  WO3-RSP-REQ-GETTIME  PIC X(8).
      *
       01  MO1-MESSAGE-CON.
           03  MO1-CON1                    PIC X(22).
           03  MO1-CON2                    PIC X(50).
       01  WO3-PUT-BUFFER.
           COPY PRGQO1X1.
```

Figure 38 (Part 10 of 40). MVS WTPING Cobol Program Source

```
      *
      *
      *    TARGET QUEUE MANAGER NAME FROM THE INPUT SCREEN
      *
       01  WO3-TGTQMGR                    PIC X(48).
       01  WO3-OTHER.
           03  WO3-PARMS.
               05  WO3-NOMSG.
                   10 FILLER PIC X(6).
                   10 WO3-NOMSG-PARM PIC X(5).
               05  WO3-LENMSG.
                   10 FILLER PIC X(7).
                   10 WO3-LENMSG-PARM PIC X(6).
               05  WO3-FBYTES.
                   10 FILLER PIC X(7).
                   10 WO3-FBYTES-PARM PIC X(10).
               05  WO3-REMQMGR.
                   10 FILLER PIC X(8).
                   10 WO3-REMQMGR-PARM PIC X(20).
               05  WO3-COMMENT.
                   10 FILLER PIC X(8).
                   10 WO3-COMMENT-PARM PIC X(50).
               05  WO3-RESPMSG.
                   10 FILLER PIC X(8).
                   10 WO3-RESPMSG-PARM PIC X(5).
      *
      *
      *    MESSAGE FOR DIAGNOSTIC TRACE
      *
       01  DIAG-MSG                   PIC X(80).
       01  DIAG-MSG-LENGTH            PIC S9(4) BINARY.
      ****************************************************************
      *
      *    WO4 - PROCESS FLAGS
      *
       01  WO4-PROCESS-FLAG.
           03  WO4-INPUT-STATUS       PIC X(3)   VALUE 'OK '.
               88  INPUT-INVALID                 VALUE 'BAD'.
               88  INPUT-OK                      VALUE 'OK '.
           03  WO4-INPUT-FLAG         PIC 9      VALUE 0.
```

Figure 38 (Part 11 of 40). MVS WTPING Cobol Program Source

```
      *
           03  WO4-FST-MSG-FLAG        PIC 9      VALUE 0.
               88  FST-MSG-PROCESSED              VALUE 1.
           03  WO4-LST-MSG-FLAG        PIC 9      VALUE 0.
               88  LST-MSG-PROCESSED              VALUE 1.
      *
           03  WO4-CALL-STATUS         PIC X(6)   VALUE 'OK'.
               88  WO4-CALL-OK                    VALUE 'OK'.
               88  WO4-CALL-FAILED                VALUE 'FAILED'.
      *
      *
      *    MO1 - ERROR MESSAGES FOR DISPLAYING
      *
       01  MO1-MESSAGE.
      *
           03  MO1-MESSAGE-1           PIC X(79) VALUE
                 'VALUE MUST BE INTEGER NUMERIC'.
      *
           03  MO1-MESSAGE-2           PIC X(79)  VALUE
                 'MESSAGE LENGTH CAN ONLY ON 10, 1000 OR 10000'.
      *
           03  MO1-MESSAGE-21          PIC X(79)  VALUE
                 'TARGET Q MANAGER NAME IS INVALID'.
      *
           03  MO1-MESSAGE-3           PIC X(79)  VALUE
                 'PRESS PF5 TO MAKE ANOTHER INQUIRY'.
      *
           03  MO1-MESSAGE-4.
               10  MO1-MSG4-OPERATION  PIC X(08)  VALUE SPACES.
               10  MO1-MSG4-OBJECTNAME PIC X(22)  VALUE SPACES.
               10  FILLER              PIC X(18)  VALUE
                                             ' ERROR: COMP CODE:'.
               10  MO1-MSG4-COMPCODE   PIC Z(08)9 VALUE ZERO.
               10  FILLER              PIC X(09)  VALUE ' REASON:' .
               10  MO1-MSG4-REASON     PIC Z(08)9 VALUE ZERO.
      *
      *
           03  MO1-MESSAGE-6                PIC X(75) VALUE
                 ' MATCHED...BUT NOT FIRST AND LAST.....       '.
      *
```

Figure 38 (Part 12 of 40). MVS WTPING Cobol Program Source

```
           03  MO1-MESSAGE-7                PIC X(75) VALUE
                ' NO MATCHING MESSAGE IN RESPONSE Q.....      '.
      *
           03  MO1-MESSAGE-8                PIC X(75) VALUE
               ' INCOMPLETE!  NO CORRESP. MSG AVAILABLE IN RESP.Q YET.'.
      *
           03  MO1-MESSAGE-9                PIC X(75) VALUE
               ' PLEASE WAIT....STILL IN PROGRESS....AT SERVER SITE...'.
      *
           03  MO1-MESSAGE-FINAL            PIC X(75) VALUE
                ' FINALLY ALL DONE !!                         '.
      *
      *
      *    API CONTROL BLOCK
      *
       01  MQM-OBJECT-DESCRIPTOR.
           COPY CMQODV.
       01  MQM-MESSAGE-DESCRIPTOR.
           COPY CMQMDV.
       01  MQM-GET-MESSAGE-OPTIONS.
           COPY CMQGMOV.
       01  MQM-PUT-MESSAGE-OPTIONS.
           COPY CMQPMOV.
      *
      *    MQV CONTAINS CONSTANTS AND RETURN CODES
      *
       01  MQM-CONST-RC.
           COPY CMQV   SUPPRESS.
      *
      * ---------------------------------------------------------- *
       EJECT
       COPY PRGTRAN.
       EJECT
       LINKAGE SECTION.
       01  PARMDATA.
           05  PARM-LEN               PIC S9(03) BINARY.
           05  PARM-STRING            PIC X(100).
      *
      * ---------------------------------------------------------- *
       PROCEDURE DIVISION USING PARMDATA.
```

Figure 38 (Part 13 of 40). MVS WTPING Cobol Program Source

```
      * ------------------------------------------------------------- *
      * ------------------------------------------------------------- *
       A-MAIN SECTION.
      * ------------------------------------------------------------- *
      *
      * THIS SECTION INTERPRETS THE PARM CARDS
      * AND SELECTS QUEUE-MANAGER
      *
      * ------------------------------------------------------------- *
       PARM-INFO-SELECT.
      *
      *    IF NO DATA WAS PASSED, CREATE A MESSAGE, PRINT IT, AND
      *    EXIT
      *
           IF PARM-LEN = 0 THEN
              STRING 'MQS95001 : 95001  '
                     'NO PARM-CARD PASSED '
                      DELIMITED BY SIZE INTO OUTMSG
              PERFORM DISPLAY-CONS
              GO TO A-MAIN-EXIT
           END-IF.
      *
      *    SEPARATE INTO THE RELEVANT FIELDS ANY DATA PASSED IN THE
      *    PARM STATEMENT
      *
           UNSTRING PARM-STRING DELIMITED BY ALL ','
                                  INTO WO2-MQM
                                       WO2-CLIENT-QNAME.
      *
      *    CHECK FOR VALID QUEUEMANAGER AND QUEUE COMBINATION
      *
           IF WO2-MQM = 'CSQ1' THEN GO TO APPL-CONNECT-QMGR.
           IF WO2-MQM = 'CSQ2' THEN GO TO APPL-CONNECT-QMGR.
           STRING 'MQS95001 : 95001           '
                  'NO VALID QUEUEMANAGER IN PARM '
                  DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
             GO   TO A-MAIN-EXIT.
       APPL-CONNECT-QMGR.
```

Figure 38 (Part 14 of 40). MVS WTPING Cobol Program Source

```
      * ------------------------------------------------------------ *
      *                                                              *
      * THIS SECTION CONNECTS THE SERVER TO QMGR                     *
      *                                                              *
      * ------------------------------------------------------------ *
           IF WO2-CLIENT-QNAME = SPACE THEN
               STRING 'MQS95001 : 95001          '
                      'NO VALID CLIENT QNAME IN PARM '
                      DELIMITED BY SIZE INTO OUTMSG
                 PERFORM DISPLAY-CONS
                 GO   TO A-MAIN-EXIT.
      *
      *
      *    CONNECT TO THE SPECIFIED QUEUE MANAGER.
      *
           CALL 'MQCONN' USING WO2-MQM
                               WO3-HCONN
                               WO3-COMPCODE
                               WO3-REASON.
      *
      *    TEST THE OUTPUT OF THE CONNECT CALL.  IF THE CALL FAILED,
      *    PRINT AN ERROR MESSAGE SHOWING THE COMPLETION CODE AND
      *    REASON CODE
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQS95001 : MQS95001 CONNECT ERROR' TO OUTMSG
              PERFORM DISPLAY-CONS
              GO   TO A-MAIN-EXIT
           END-IF.
           IF WO2-MQM = 'CSQ1' THEN
             STRING 'MQS95001 : 95001 SA25 CSQ1 '
                      'CLIENT CONNECTED '
                     DELIMITED BY SIZE INTO OUTMSG
             MOVE 'THIS PROGRAM USES A DYNAMIC REPLY-Q' TO OUTMSG
             PERFORM DISPLAY-LOCAL
            ELSE
             STRING 'MQS95001 : 95001 SA18 CSQ2 '
                      'CLIENT CONNECTED '
                     DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
```

Figure 38 (Part 15 of 40). MVS WTPING Cobol Program Source

```
            MOVE 'THIS PROGRAM USES A DYNAMIC REPLY-Q' TO OUTMSG
            PERFORM DISPLAY-LOCAL
           END-IF.
       MAIN-PROCESS SECTION.
      *
      *
      *   GET  PARMIN MESSAGE
      *
      *
       READ-INI-MSG.
           IF WO3-REQUESTS > WO3-MAX-REQUESTS THEN
              DISPLAY 'MAX NUMBER OF REQUESTS REACHED'
              DISPLAY 'MAX NUMBER = ' WO3-MAX-REQUESTS
              DISPLAY 'WEB CLIENT PROCESSOR TERMINATED'
              GO TO A-MAIN-EXIT
           END-IF.
           PERFORM GET-CLIENT-PARMS.
      *
      *
      *
      *
      *-------------------------------------------------------------*
      *
      *    GET     CURRENT-DATE
      *
      *
      *-------------------------------------------------------------*
           MOVE    SPACES         TO  WOO-CURRENT-DATE.
           PERFORM ASK-DATETIME.
           MOVE    WOO-TEMP-YEAR  TO  WOO-CUR-YYYY.
           MOVE    WOO-TEMP-MM    TO  WOO-CUR-MM.
           MOVE    WOO-TEMP-DD    TO  WOO-CUR-DD.
      *
           MOVE SPACE        TO  WOO-MESSAGE.
           SET  WO4-CALL-OK  TO  TRUE.
           MOVE 0            TO  WO4-FST-MSG-FLAG.
           MOVE 0            TO  WO4-LST-MSG-FLAG.
           PERFORM VALIDATE-INPUT.
           IF NOT INPUT-OK
                     MOVE WOO-MESSAGE TO WOO-MESSAGE-OUT
```

Figure 38 (Part 16 of 40). MVS WTPING Cobol Program Source

```
                    MOVE WOO-MESSAGE-OUT TO OUTMSG
                    PERFORM DISPLAY-CONS
                    GO TO A-MAIN-EXIT
          END-IF.
          PERFORM OPEN-QUEUES.
          PERFORM PROCESS-REQUEST.
          IF WOO-MESSAGE NOT = SPACES
             MOVE WOO-MESSAGE TO WOO-MESSAGE-OUT
             MOVE WOO-MESSAGE-OUT TO OUTMSG
             PERFORM DISPLAY-CONS
           ELSE
             MOVE MO1-MESSAGE-FINAL TO WOO-MESSAGE-OUT
             MOVE WOO-MESSAGE-OUT TO OUTMSG
             PERFORM DISPLAY-CONS
          END-IF.
          PERFORM CLOSE-QUEUES.
          PERFORM PUT-MSG-TO-CLIENT.
          GO TO READ-INI-MSG.
       A-MAIN-EXIT.
          GOBACK.
          EJECT
      *
       VALIDATE-INPUT SECTION.
      *---------------------------------------------------------------*
      *
      *    VALIDATE THE USER INPUT
      *
      *---------------------------------------------------------------*
          EVALUATE TRUE
              WHEN WOO-MSG-TOT    NOT NUMERIC    OR
                   WOO-MSG-LENGTH NOT NUMERIC    OR
                   WOO-GETRSP     NOT NUMERIC
                   MOVE MO1-MESSAGE-1 TO WOO-MESSAGE
                   MOVE 1 TO WO4-INPUT-FLAG
              WHEN OTHER
                   EVALUATE TRUE
                       WHEN WOO-MSG-LENGTH  NOT = 10      AND
                            WOO-MSG-LENGTH  NOT = 100     AND
                            WOO-MSG-LENGTH  NOT = 1000    AND
                            WOO-MSG-LENGTH  NOT = 10000
```

Figure 38 (Part 17 of 40). MVS WTPING Cobol Program Source

```
                         MOVE MO1-MESSAGE-2 TO WOO-MESSAGE
                         MOVE 1 TO WO4-INPUT-FLAG
                    WHEN OTHER
                         MOVE O TO WO4-INPUT-FLAG
               END-EVALUATE
          END-EVALUATE
      *
          IF WO4-INPUT-FLAG NOT = O
             SET  INPUT-INVALID TO TRUE
             GO TO VALIDATE-INPUT-EXIT
          END-IF
      *
      *
      *
      *
          EVALUATE TRUE
             WHEN WO3-TGTQMGR = QMGR-CSQ1
               MOVE 'DXB.CSQ1.REQUESTQ                              '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-CSQ2
               MOVE 'DXB.CSQ2.REQUESTQ                              '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-MQOS1
               MOVE 'PHD.MQOS1.REQUESTQ                             '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-MQOS2
               MOVE 'PHD.MQOS2.REQUESTQ                             '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-MQOS3
               MOVE 'PHD.MQOS3.REQUESTQ                             '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-RS60001
               MOVE 'PHD.RS60001.REQUESTQ                           '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-RS60002
               MOVE 'PHD.RS60002.REQUESTQ                           '
               TO WO2-REQ-Q
             WHEN WO3-TGTQMGR = QMGR-RS60003
               MOVE 'PHD.RS60003.REQUESTQ                           '
               TO WO2-REQ-Q
```

Figure 38 (Part 18 of 40). MVS WTPING Cobol Program Source

```
           WHEN WO3-TGTQMGR = QMGR-RS600010
             MOVE 'PHD.RS600010.REQUESTQ                          '
             TO WO2-REQ-Q
           WHEN WO3-TGTQMGR = QMGR-DEFAULT
             MOVE 'PHD.DEFAULT.REQUESTQ                           '
             TO WO2-REQ-Q
           WHEN OTHER
             MOVE MO1-MESSAGE-21 TO WOO-MESSAGE
             MOVE 1 TO WO4-INPUT-FLAG
       END-EVALUATE
     *
       IF WO4-INPUT-FLAG = 1
            SET  INPUT-INVALID TO TRUE
       ELSE
            SET  INPUT-OK      TO TRUE
       END-IF.
      VALIDATE-INPUT-EXIT.
       EXIT.
       EJECT
     *
     *
      OPEN-QUEUES SECTION.
     *-------------------------------------------------------------*
     *
     * OPEN INPUT AND RESPONSE QUEUES
     * IF THE OPEN FAILS, A MESSAGE IS BUILT INDICATING THE REASON
     *
     *-------------------------------------------------------------*
     *
     * INITIALIZE THE OBJECT DESCRIPTION (MQOD) CONTROL BLOCK
     *
       MOVE MQOT-Q            TO  MQOD-OBJECTTYPE
       MOVE WO2-REQ-Q         TO  MQOD-OBJECTNAME
     *
       COMPUTE WO3-OPTIONS = MQOO-OUTPUT
     *
       MOVE ZERO  TO  WO3-HOBJ-REQUESTQ
     *
       CALL 'MQOPEN'  USING WO3-HCONN
                            MQOD
                            WO3-OPTIONS
                            WO3-HOBJ-REQUESTQ
                            WO3-COMPCODE
                            WO3-REASON.
```

Figure 38 (Part 19 of 40). MVS WTPING Cobol Program Source

```
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'         TO  MO1-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
              MOVE WO3-COMPCODE    TO  MO1-MSG4-COMPCODE
              MOVE WO3-REASON      TO  MO1-MSG4-REASON
              MOVE MO1-MESSAGE-4   TO  WOO-MESSAGE
              MOVE WOO-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              DISPLAY 'ERROR ON REQUESTQ'
              GO   TO A-MAIN-EXIT
           END-IF
      *
      *       MOVE MQOD-OBJECTQMGRNAME TO  KD-QMGR
              MOVE '        ' TO  KD-QMGR
      *
      * INITIALIZE THE OBJECT DESCRIPTION (MQOD) CONTROL BLOCK
      *
      *
           MOVE MQOT-Q       TO  MQOD-OBJECTTYPE
      *    TELL QMGR TO GENERATE QNAME WITH PREFIX
           MOVE WO2-MODEL-Q  TO  MQOD-OBJECTNAME
           MOVE 'WTPONG.*'   TO  MQOD-DYNAMICQNAME
      *
           COMPUTE WO3-OPTIONS = MQOO-INPUT-SHARED      +
                                 MQOO-SAVE-ALL-CONTEXT
      *
           MOVE ZERO  TO  WO3-HOBJ-RESPQ
      *
           CALL 'MQOPEN'  USING WO3-HCONN
                                MQOD
                                WO3-OPTIONS
                                WO3-HOBJ-RESPQ
                                WO3-COMPCODE
                                WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'         TO  MO1-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
              MOVE WO3-COMPCODE    TO  MO1-MSG4-COMPCODE
```

Figure 38 (Part 20 of 40). MVS WTPING Cobol Program Source

```
            MOVE WO3-REASON      TO  MO1-MSG4-REASON
            MOVE MO1-MESSAGE-4   TO  WOO-MESSAGE
            MOVE WOO-MESSAGE TO OUTMSG
            PERFORM DISPLAY-CONS
            DISPLAY 'MQOPEN ERROR ON DYNAMIC REPLY QUEUE'
            GO   TO A-MAIN-EXIT
         END-IF.
      *
      *
      *    SAVE QMGR GENERATED REPLY QUEUE NAME
      *
      *
         MOVE MQOD-OBJECTNAME TO WO2-RESP-Q
            STRING 'PROGRAM USES DYN-QUEUE '
                    MQOD-OBJECTNAME
                    DELIMITED BY SIZE INTO OUTMSG
            PERFORM DISPLAY-LOCAL.
       OPEN-QUEUES-EXIT.
         EXIT.
         EJECT
      *
      *
      *
       CLOSE-QUEUES  SECTION.
      *-----------------------------------------------------------------*
      *
      *    CLOSE ALL QUEUE BEFORE TERMINATING
      *
      *-----------------------------------------------------------------*
         CALL 'MQCLOSE' USING WO3-HCONN
                              WO3-HOBJ-REQUESTQ
                              MQCO-NONE
                              WO3-COMPCODE
                              WO3-REASON.
      *
         IF WO3-COMPCODE NOT = MQCC-OK
            MOVE 'MQCLOSE'         TO  MO1-MSG4-OPERATION
            MOVE MQOD-OBJECTNAME   TO  MO1-MSG4-OBJECTNAME
            MOVE WO3-COMPCODE      TO  MO1-MSG4-COMPCODE
            MOVE WO3-REASON        TO  MO1-MSG4-REASON
```

Figure 38 (Part 21 of 40). MVS WTPING Cobol Program Source

```
           MOVE MO1-MESSAGE-4    TO  WOO-MESSAGE
           MOVE WOO-MESSAGE TO OUTMSG
           PERFORM DISPLAY-CONS
        END-IF.
      *
      *
      *    MQCO_DELETE DELETE PURGED TEMP DYNAMIC QUEUE
      *
      *
        CALL 'MQCLOSE' USING WO3-HCONN
                             WO3-HOBJ-RESPQ
                             MQCO-DELETE
                             WO3-COMPCODE
                             WO3-REASON.
      *
        IF WO3-COMPCODE NOT = MQCC-OK
           MOVE 'MQCLOSE'         TO  MO1-MSG4-OPERATION
           MOVE MQOD-OBJECTNAME   TO  MO1-MSG4-OBJECTNAME
           MOVE WO3-COMPCODE      TO  MO1-MSG4-COMPCODE
           MOVE WO3-REASON        TO  MO1-MSG4-REASON
           MOVE MO1-MESSAGE-4     TO  WOO-MESSAGE
        END-IF.
      *
       CLOSE-QUEUES-EXIT.
        EXIT.
        EJECT
      *
      *
      *
      *
       PROCESS-REQUEST SECTION.
      *-------------------------------------------------------------*
      *
      * PROCESS THE USER REQUESTS :
      *  - BUILD MESSAGE
      *  - SEND 1ST MESSAGE, TAKE TIMESTAMP AND ACKNOWLEDGE THE USER
      *  - SEND THE REST OF THE MESSAGE UNTIL ALL MESSAGE SENT
      *       - TAKE TIMESTAMP
      *       - ACKNOWLEDGE THE USER FOR EVERY NN MESSAGES PROCESSED,
      *         WHERE NN IS THE VALUE SUPPLIED BY THE USER
```

Figure 38 (Part 22 of 40). MVS WTPING Cobol Program Source

```
*       - IF LAST MSG
*           ACKNOWLEDGE THE USER WITH A HAPPY-MSG
*
*-----------------------------------------------------------------*
*
*    BUILD THE MESSAGE
*
     MOVE SPACES    TO  WO3-PUT-BUFFER.
     MOVE 0         TO  WOO-MSG-SEQ     WOO-MSG-PROC
     MOVE 0         TO  WOO-MSG-NO-PROC WOO-CNT-1  WOO-MSG-CNT.
     MOVE WOO-MSG-TOT TO  WO3-REQ-MSG-TOT TOT-MSG.
     MOVE WOO-MSG-LENGTH  TO  WO3-REQ-MSG-LEN.
     MOVE WOO-COMMENT  TO  WO3-REQ-MSG-COMMENT.
*
     IF WOO-MSG-LENGTH = 10
        MOVE WOO-FIRSTTXT TO WO3-REQ-MSG
     ELSE
       IF WOO-MSG-LENGTH = 1000
          MOVE    100   TO  WOO-MSG-CNT
          MOVE    0     TO  WOO-CNT-1
          PERFORM BUILD-MESSAGE WITH TEST BEFORE
                                UNTIL WOO-CNT-1 = WOO-MSG-CNT
       ELSE
          MOVE    1000  TO  WOO-MSG-CNT
          MOVE    0     TO  WOO-CNT-1
          PERFORM BUILD-MESSAGE WITH TEST BEFORE
                                UNTIL WOO-CNT-1 = WOO-MSG-CNT
       END-IF
     END-IF.
*
*
* LOOP FROM HERE TO END-PERFORM UNTIL THE LAST MESSAGE SENT
*                                     OR ANY MQI CALL FAILED
*
     PERFORM WITH TEST BEFORE UNTIL  WOO-MSG-SEQ = WOO-MSG-TOT OR
                                     WO4-CALL-FAILED
        ADD  1              TO  WOO-MSG-SEQ
        ADD  1              TO  WOO-MSG-PROC
        MOVE WO2-RESP-Q     TO  MQMD-REPLYTOQ
        MOVE WO2-MQM        TO  MQMD-REPLYTOQMGR
```

Figure 38 (Part 23 of 40). MVS WTPING Cobol Program Source

```
      *
              IF   WOO-MSG-SEQ = 1
                   PERFORM ASK-DATETIME
                   MOVE 'FIRST   '         TO  MQMD-MSGID
                   MOVE WOO-TEMP-DATETIME TO  MQMD-CORRELID
                   MOVE WOO-TEMP-PUTTIME  TO  KD-TIST
                   MOVE MQMT-REQUEST      TO  MQMD-MSGTYPE
              ELSE
                IF WOO-MSG-SEQ = WOO-MSG-TOT
                   MOVE 'LAST    '         TO  MQMD-MSGID
                   MOVE WOO-FST-CORRELID  TO  MQMD-CORRELID
                   MOVE MQMT-REQUEST      TO  MQMD-MSGTYPE
                ELSE
                   MOVE 'NOTFIRST'         TO  MQMD-MSGID
                   MOVE WOO-TEMP-PUTTIME  TO  KD-TIST
                   MOVE WOO-FST-CORRELID  TO  MQMD-CORRELID
                   MOVE MQMT-DATAGRAM     TO  MQMD-MSGTYPE
                END-IF
              END-IF
      *
              COMPUTE MQPMO-OPTIONS = MQPMO-NO-SYNCPOINT
      *
              COMPUTE WO3-DATALEN  =
                   LENGTH OF WO3-REQUEST +  WOO-MSG-LENGTH
      *
              MOVE MQCCSI-Q-MGR TO MQMD-CODEDCHARSETID
      *
      * SPECIFY THE FORMAT AS STRING
      *
              MOVE MQFMT-STRING TO MQMD-FORMAT

              CALL 'MQPUT'  USING  WO3-HCONN
                                   WO3-HOBJ-REQUESTQ
                                   MQMD
                                   MQPMO
                                   WO3-DATALEN
                                   WO3-PUT-BUFFER
                                   WO3-COMPCODE
                                   WO3-REASON
      *
```

Figure 38 (Part 24 of 40). MVS WTPING Cobol Program Source

```
      * TEST THE OUTCOME OF THE CALL
      *   IF THE MESSAGE WAS SENT SUCCESSFULLY
      *      IDENTIFY THE QUEUE AND QUEUE MANAGER WHICH RECEIVED THE
      *               MESSAGE
      *   OTHERWISE
      *      SET AN APPROPRIATE ERROR MESSAGE
      *
             EVALUATE TRUE
                 WHEN WO3-COMPCODE = MQCC-OK
      **              PERFORM ASK-DATETIME
                      MOVE MQMD-PUTDATE  TO WOO-MQMD-PUTDATE
                      MOVE MQMD-PUTTIME  TO WOO-MQMD-PUTTIME
                      PERFORM GMT-TO-CURTIME
                      MOVE WOO-PUT-YYYY  TO WOO-TEMP-PUT-YYYY
                      MOVE WOO-PUT-MM    TO WOO-TEMP-PUT-MM
                      MOVE WOO-PUT-DD    TO WOO-TEMP-PUT-DD
                      MOVE WOO-PUT-HR    TO WOO-TEMP-PUT-HR
                      MOVE WOO-PUT-MIN   TO WOO-TEMP-PUT-MIN
                      MOVE WOO-PUT-SEC   TO WOO-TEMP-PUT-SEC
                      MOVE WOO-PUT-TH    TO WOO-TEMP-PUT-TH
      *
                 WHEN WO3-COMPCODE NOT = MQCC-OK
                      MOVE 'MQPUT'             TO MO1-MSG4-OPERATION
                      MOVE MQOD-OBJECTNAME     TO MO1-MSG4-OBJECTNAME
                      MOVE WO3-COMPCODE        TO MO1-MSG4-COMPCODE
                      MOVE WO3-REASON          TO MO1-MSG4-REASON
                      MOVE MO1-MESSAGE-4       TO WOO-MESSAGE
                      MOVE WOO-MESSAGE TO OUTMSG
                      PERFORM DISPLAY-CONS
                      SET  WO4-CALL-FAILED     TO TRUE
                      GO   TO PROCESS-REQUEST-EXIT
             END-EVALUATE
      *
             IF WOO-MSG-SEQ = 1
      **        MOVE    WOO-TEMP-DATE  TO  WOO-FSTREQ-PUTDATE
      **        MOVE    WOO-TEMP-TIME  TO  WOO-FSTREQ-PUTTIME
                MOVE    WOO-TEMP-PUTDATE  TO  WOO-FSTREQ-PUTDATE
                MOVE    WOO-TEMP-PUTTIME  TO  WOO-FSTREQ-PUTTIME
                MOVE    MQMD-MSGID        TO  WOO-FST-MSGID
                MOVE    MQMD-CORRELID     TO  WOO-FST-CORRELID
```

Figure 38 (Part 25 of 40). MVS WTPING Cobol Program Source

```
               MOVE FSTREQ TO OUTMSG
               PERFORM DISPLAY-LOCAL
               PERFORM GET-RESPQ-MSG
            ELSE
               IF WOO-MSG-SEQ = WOO-MSG-TOT
                  ADD     WOO-MSG-PROC    TO  WOO-MSG-NO-PROC
                  MOVE    0               TO  WOO-MSG-PROC
      **          MOVE    WOO-TEMP-DATE   TO  WOO-LSTREQ-PUTDATE
      **          MOVE    WOO-TEMP-TIME   TO  WOO-LSTREQ-PUTTIME
                  MOVE    WOO-TEMP-PUTDATE  TO  WOO-LSTREQ-PUTDATE
                  MOVE    WOO-TEMP-PUTTIME  TO  WOO-LSTREQ-PUTTIME
                  MOVE    WOO-MSG-NO-PROC   TO  NO-PROC
                  MOVE    MQMD-MSGID        TO  WOO-LST-MSGID
                  MOVE    MQMD-CORRELID     TO  WOO-LST-CORRELID
                  MOVE LSTREQ TO OUTMSG
                  PERFORM DISPLAY-LOCAL
                  PERFORM GET-RESPQ-MSG
               ELSE
                  IF WOO-MSG-PROC = WOO-GETRSP
                     ADD     WOO-MSG-PROC    TO  WOO-MSG-NO-PROC
                     MOVE    0               TO  WOO-MSG-PROC
                     MOVE    WOO-MSG-NO-PROC TO  NO-PROC
                  ELSE
                     IF  NOT FST-MSG-PROCESSED
                         PERFORM GET-RESPQ-MSG
                     END-IF
                  END-IF
               END-IF
            END-IF
      *
           END-PERFORM.
      *
           IF WOO-MESSAGE NOT = SPACE
              GO TO PROCESS-REQUEST-EXIT.
      *
      *    LOOP FROM HERE UNTIL LAST MEASSGE FROM RESPONSE QUEUE
      *         IS PROCESSED OR LOOP-CNT = 500
      *    DISPLAY ERROR MESSAGE IF PROCESSING IS INCOMPLETE
      *
           MOVE  0 TO  WOO-LOOP-CNT
```

Figure 38 (Part 26 of 40). MVS WTPING Cobol Program Source

```
           PERFORM GET-RESPQ-MSG WITH TEST BEFORE
                   UNTIL (FST-MSG-PROCESSED  AND  LST-MSG-PROCESSED) OR
                          W00-LOOP-CNT = 500.
      *
      *    LOOP FOR ANOTHER ROUND IF PROCESSING IS NOT YET COMPLETED
      *    BUT FIRST WAIT FOR SOME TIME
      *
           IF NOT (FST-MSG-PROCESSED  AND  LST-MSG-PROCESSED)
      * SEND A MESSAGE TO SAY SORRY FOR THE DELAY
              MOVE M01-MESSAGE-9  TO  W00-MESSAGE
              MOVE W00-MESSAGE TO W00-MESSAGE-OUT
              DISPLAY W00-MESSAGE-OUT
      * NOW RETRY
              MOVE  0 TO  W00-LOOP-CNT
              PERFORM WITH TEST BEFORE
                   UNTIL (FST-MSG-PROCESSED  AND  LST-MSG-PROCESSED) OR
                          W00-LOOP-CNT = 60
      * BUT BEFORE RETRYING, GMO-WAIT 1 SEC
                PERFORM GET-RESPQ-MSG1
              END-PERFORM
           END-IF
      *
           IF NOT (FST-MSG-PROCESSED  AND  LST-MSG-PROCESSED)
              MOVE M01-MESSAGE-8  TO  W00-MESSAGE
           ELSE
              MOVE SPACES TO  W00-MESSAGE
           END-IF.

      *
       PROCESS-REQUEST-EXIT.
           EXIT.
           EJECT
      *
      *
      *-----------------------------------------------------------------*
       ASK-DATETIME SECTION.
      *-----------------------------------------------------------------*
      *
      *
           ACCEPT W05-DATE FROM DATE.
```

Figure 38 (Part 27 of 40). MVS WTPING Cobol Program Source

```
           ACCEPT WO5-TIME FROM TIME.
           STRING  WO5-DD WO5-DATESEP WO5-MM WO5-DATESEP
                    WO5-YY
                   DELIMITED BY SIZE INTO WOO-TEMP-DATE.
           STRING  WO5-HH WO5-TIMESEP WO5-MI WO5-TIMESEP
                   WO5-SS
                   DELIMITED BY SIZE INTO WOO-TEMP-TIME.
           STRING  '19' WO5-YY
                   DELIMITED BY SIZE INTO WO5-YEAR.
           MOVE WO5-YEAR TO WOO-TEMP-YEAR.
      *
       ASK-DATETIME-EXIT.
           EXIT.
           EJECT
      *
      *-----------------------------------------------------------------*
       BUILD-MESSAGE SECTION.
      *-----------------------------------------------------------------*
      *
      *    BUILD MESSAGE TO BE SENT ....
      *
           ADD  1         TO  WOO-CNT-1
           MOVE WOO-FIRSTTXT TO  WO3-REQ-MSG-SEG(WOO-CNT-1).
      *
       BUILD-MESSAGE-EXIT.
           EXIT.
           EJECT
      *
      *
      *-----------------------------------------------------------------*
       GET-RESPQ-MSG  SECTION.
      *-----------------------------------------------------------------*
      *
      *  GET MESSAGE FROM THE RESPONSE QUEUE
      *
           IF NOT FST-MSG-PROCESSED
              MOVE WOO-FST-MSGID    TO  MQMD-MSGID
              MOVE WOO-FST-CORRELID TO  MQMD-CORRELID
           ELSE
              MOVE WOO-LST-MSGID    TO  MQMD-MSGID
```

Figure 38 (Part 28 of 40). MVS WTPING Cobol Program Source

```
           MOVE WOO-LST-CORRELID TO  MQMD-CORRELID.
      *
           ADD      1          TO   WOO-LOOP-CNT.
      *
           COMPUTE  MQGMO-OPTIONS = MQGMO-NO-SYNCPOINT     +
                                    MQGMO-NO-WAIT +
                                    MQGMO-ACCEPT-TRUNCATED-MSG
      *
           MOVE LENGTH OF WO3-GET-BUFFER  TO  WO3-BUFFLEN.
      *
           CALL 'MQGET' USING WO3-HCONN
                              WO3-HOBJ-RESPQ
                              MQMD
                              MQGMO
                              WO3-BUFFLEN
                              WO3-GET-BUFFER
                              WO3-DATALEN
                              WO3-COMPCODE
                              WO3-REASON.
           EVALUATE TRUE
               WHEN (WO3-COMPCODE = MQCC-OK  AND
                    WO3-REASON   = MQRC-NONE) OR
                    WO3-REASON   = MQRC-TRUNCATED-MSG-ACCEPTED

                 IF MQMD-CODEDCHARSETID = 500
                    MOVE MQMD-PUTDATE  TO WOO-MQMD-PUTDATE
                    MOVE MQMD-PUTTIME  TO WOO-MQMD-PUTTIME
                    PERFORM GMT-TO-CURTIME
                    MOVE WOO-PUT-YYYY  TO WOO-TEMP-PUT-YYYY
                    MOVE WOO-PUT-MM    TO WOO-TEMP-PUT-MM
                    MOVE WOO-PUT-DD    TO WOO-TEMP-PUT-DD
                    MOVE WOO-PUT-HR    TO WOO-TEMP-PUT-HR
                    MOVE WOO-PUT-MIN   TO WOO-TEMP-PUT-MIN
                    MOVE WOO-PUT-SEC   TO WOO-TEMP-PUT-SEC
                    MOVE WOO-PUT-TH    TO WOO-TEMP-PUT-TH
                 ELSE
                    MOVE '?ASCHI?' TO WO3-RSP-REQ-GETDATE
                    MOVE '?ASCHI?' TO WO3-RSP-REQ-GETTIME
                 END-IF
                    MOVE MQMD-MSGID    TO WO3-RESP-MSGID
```

Figure 38 (Part 29 of 40). MVS WTPING Cobol Program Source

```
                MOVE MQMD-CORRELID TO WO3-RESP-CORRELID
                PERFORM PROCESS-RESPONSE-MESSAGE
                SET     WO4-CALL-OK  TO  TRUE
      *
              WHEN WO3-COMPCODE = MQCC-FAILED   AND
                   WO3-REASON   = MQRC-NO-MSG-AVAILABLE
                   SET     WO4-CALL-OK  TO  TRUE
      *
              WHEN OTHER
                   MOVE 'MQGET RESP'    TO  MO1-MSG4-OPERATION
                   MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
                   MOVE WO3-COMPCODE    TO  MO1-MSG4-COMPCODE
                   MOVE WO3-REASON      TO  MO1-MSG4-REASON
                   MOVE MO1-MESSAGE-4   TO  WOO-MESSAGE
                   SET  WO4-CALL-FAILED TO  TRUE
      *
           END-EVALUATE.
      *
       GET-RESPQ-MSG-EXIT.
           EXIT.
      *-------------------------------------------------------------*
       GET-RESPQ-MSG1 SECTION.
      *-------------------------------------------------------------*
      *
      *  GET MESSAGE FROM THE RESPONSE QUEUE
      *
           IF NOT FST-MSG-PROCESSED
              MOVE WOO-FST-MSGID    TO  MQMD-MSGID
              MOVE WOO-FST-CORRELID TO  MQMD-CORRELID
           ELSE
              MOVE WOO-LST-MSGID    TO  MQMD-MSGID
              MOVE WOO-LST-CORRELID TO  MQMD-CORRELID.
      *
           ADD      1          TO   WOO-LOOP-CNT.
      *
           COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                                   MQGMO-ACCEPT-TRUNCATED-MSG +
                                   MQGMO-NO-SYNCPOINT.
      *
      *  INSERT WAIT-INTERVAL 1 SEC PER LOOP
```

Figure 38 (Part 30 of 40). MVS WTPING Cobol Program Source

```
      *
           MOVE +1000              TO MQGMO-WAITINTERVAL
      *
           MOVE LENGTH OF WO3-GET-BUFFER  TO  WO3-BUFFLEN.
      *
           CALL 'MQGET' USING WO3-HCONN
                              WO3-HOBJ-RESPQ
                              MQMD
                              MQGMO
                              WO3-BUFFLEN
                              WO3-GET-BUFFER
                              WO3-DATALEN
                              WO3-COMPCODE
                              WO3-REASON.
           EVALUATE TRUE
               WHEN (WO3-COMPCODE = MQCC-OK  AND
                     WO3-REASON   = MQRC-NONE) OR
                     WO3-REASON   = MQRC-TRUNCATED-MSG-ACCEPTED

                 IF MQMD-CODEDCHARSETID = 500
                    MOVE MQMD-PUTDATE  TO WOO-MQMD-PUTDATE
                    MOVE MQMD-PUTTIME  TO WOO-MQMD-PUTTIME
                    PERFORM GMT-TO-CURTIME
                    MOVE WOO-PUT-YYYY  TO WOO-TEMP-PUT-YYYY
                    MOVE WOO-PUT-MM    TO WOO-TEMP-PUT-MM
                    MOVE WOO-PUT-DD    TO WOO-TEMP-PUT-DD
                    MOVE WOO-PUT-HR    TO WOO-TEMP-PUT-HR
                    MOVE WOO-PUT-MIN   TO WOO-TEMP-PUT-MIN
                    MOVE WOO-PUT-SEC   TO WOO-TEMP-PUT-SEC
                    MOVE WOO-PUT-TH    TO WOO-TEMP-PUT-TH
                 ELSE
                    MOVE '?ASCHI?' TO WO3-RSP-REQ-GETDATE
                    MOVE '?ASCHI?' TO WO3-RSP-REQ-GETTIME
                 END-IF
                    MOVE MQMD-MSGID    TO WO3-RESP-MSGID
                    MOVE MQMD-CORRELID TO WO3-RESP-CORRELID
                    PERFORM PROCESS-RESPONSE-MESSAGE
                    SET     WO4-CALL-OK  TO  TRUE
      *
               WHEN WO3-COMPCODE = MQCC-FAILED   AND
```

Figure 38 (Part 31 of 40). MVS WTPING Cobol Program Source

```
                WO3-REASON   = MQRC-NO-MSG-AVAILABLE
                SET     WO4-CALL-OK  TO  TRUE
      *
              WHEN OTHER
                MOVE 'MQGET RESP'    TO  MO1-MSG4-OPERATION
                MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
                MOVE WO3-COMPCODE    TO  MO1-MSG4-COMPCODE
                MOVE WO3-REASON      TO  MO1-MSG4-REASON
                MOVE MO1-MESSAGE-4   TO  WOO-MESSAGE
                SET  WO4-CALL-FAILED TO  TRUE
      *
           END-EVALUATE.
      *
       GET-RESPQ-MSG1-EXIT.
           EXIT.
           EJECT
      *
      *
      *-----------------------------------------------------------*
       PROCESS-RESPONSE-MESSAGE SECTION.
      *-----------------------------------------------------------*
      *
      *    PROCESS MESSAGE FROM THE RESPONSE QUEUE.
      *    DISPLAY TIMESTAMP FOR PUT AND GET TIME
      *-----------------------------------------------------------*
      *
           IF WO3-RESP-CORRELID = WOO-FST-CORRELID
              PERFORM ASK-DATETIME
              IF WO3-RESP-FIRST-MSG
                 MOVE '+MJS8 GET REPLY FIRST='  TO MO1-CON1
                 MOVE WO3-GET-BUFFER            TO MO1-CON2
                 MOVE    ' GET  '             TO  WOO-FSTREQ-GETTEXT
                 MOVE    WO3-RSP-REQ-GETDATE  TO  WOO-FSTREQ-GETDATE
                 MOVE    WOO-CUR-YYYY         TO  WOO-FSTREQ-GYR
                 MOVE    WO3-RSP-REQ-GETTIME  TO  WOO-FSTREQ-GETTIME
                 MOVE    WOO-TEMP-DATE        TO  WOO-FSTREP-GETDATE
                 MOVE    WOO-CUR-YYYY         TO  WOO-FSTREP-GYR
                 MOVE    WOO-TEMP-TIME        TO  WOO-FSTREP-GETTIME
                 MOVE    WOO-TEMP-PUTDATE     TO  WOO-FSTREP-PUTDATE
                 MOVE    WOO-TEMP-PUTTIME     TO  WOO-FSTREP-PUTTIME
```

Figure 38 (Part 32 of 40). MVS WTPING Cobol Program Source

```
               MOVE    FSTREP               TO  OUTMSG
               MOVE    FSTREP               TO  WO3-REPLY-FSTREP
               PERFORM DISPLAY-LOCAL
               SET     FST-MSG-PROCESSED    TO  TRUE
            ELSE
               IF WO3-RESP-LAST-MSG
                  MOVE '+MJS8 GET REPLY LAST='  TO MO1-CON1
                  MOVE WO3-GET-BUFFER           TO MO1-CON2
                  MOVE MO1-MESSAGE-CON TO OUTMSG
                  PERFORM DISPLAY-LOCAL
                  MOVE    ' GET  '             TO  WOO-LSTREQ-GETTEXT
                  MOVE    WO3-RSP-REQ-GETDATE TO  WOO-LSTREQ-GETDATE
                  MOVE    WOO-CUR-YYYY         TO  WOO-LSTREQ-GYR
                  MOVE    WO3-RSP-REQ-GETTIME TO  WOO-LSTREQ-GETTIME
                  MOVE    LSTREQ               TO  OUTMSG
                  PERFORM DISPLAY-LOCAL
                  MOVE    WOO-TEMP-DATE        TO  WOO-LSTREP-GETDATE
                  MOVE    WOO-CUR-YYYY         TO  WOO-LSTREP-GYR
                  MOVE    WOO-TEMP-TIME        TO  WOO-LSTREP-GETTIME
                  MOVE    WOO-TEMP-PUTDATE     TO  WOO-LSTREP-PUTDATE
                  MOVE    WOO-TEMP-PUTTIME     TO  WOO-LSTREP-PUTTIME
                  MOVE    LSTREP               TO  OUTMSG
                  MOVE    LSTREP               TO  WO3-REPLY-LSTREP
                  MOVE    MSG-PROC             TO  WO3-REPLY-NOPROC
                  PERFORM DISPLAY-LOCAL
                  SET     LST-MSG-PROCESSED    TO  TRUE
               ELSE
                  SET  WO4-CALL-FAILED  TO  TRUE
                  MOVE MO1-MESSAGE-6    TO  WOO-MESSAGE
               END-IF
            END-IF
         ELSE
            SET  WO4-CALL-FAILED  TO  TRUE
            MOVE MO1-MESSAGE-7    TO  WOO-MESSAGE
         END-IF.
      *
       PROCESS-RESPONSE-MESSAGE-EXIT.
         EXIT.
         EJECT
      *
```

Figure 38 (Part 33 of 40). MVS WTPING Cobol Program Source

```
      *
      *-------------------------------------------------------------*
       GMT-TO-CURTIME SECTION.
      *-------------------------------------------------------------*
           IF WOO-PUT-HR-NUM >= WOO-GMT-DIFF
              COMPUTE WOO-PUT-HR-NUM = WOO-PUT-HR-NUM - WOO-GMT-DIFF
           ELSE
              COMPUTE WOO-PUT-HR-NUM = 24 + WOO-PUT-HR-NUM -
                                       WOO-GMT-DIFF
              MOVE    WOO-CUR-YYYY  TO  WOO-PUT-YYYY
              MOVE    WOO-CUR-MM    TO  WOO-PUT-MM
              MOVE    WOO-CUR-DD    TO  WOO-PUT-DD.
      *
       GMT-TO-CURTIME-EXIT.
           EXIT.
           EJECT
       DISPLAY-LOCAL  SECTION.
      * ------------------------------------------------------------ *
      *    DISPLAY MESSAGE ON SYSOUT ONLY ON DEBUG MODE
      * ------------------------------------------------------------ *
      *
             DISPLAY    OUTMSG.
             MOVE SPACE TO OUTMSG.
      *
       DISPLAY-LOCAL-EXIT.
           EXIT.
       DISPLAY-CONS   SECTION.
      * ------------------------------------------------------------ *
      *    DISPLAY MESSAGE TO CONSOLE
      * ------------------------------------------------------------ *
      *
           DISPLAY    OUTMSG UPON CONSOLE.
           MOVE SPACE TO OUTMSG.
      *
       DISPLAY-CONS-EXIT.
           EXIT.
           EJECT
       GET-CLIENT-PARMS SECTION.
      * ------------------------------------------------------------ *
      *    GET PARM MESSAGE FROM WEB CLIENT
```

Figure 38 (Part 34 of 40). MVS WTPING Cobol Program Source

```
      * ------------------------------------------------------------ *
           ADD 1 TO WO3-REQUESTS.
           DISPLAY  'NO OF REQUESTS ' WO3-REQUESTS.
      *
           MOVE MQOT-Q            TO MQOD-OBJECTTYPE.
           MOVE WO2-CLIENT-QNAME  TO MQOD-OBJECTNAME.
           MOVE ZERO TO WO3-HOBJ-CLIENTQ.
      *
           MOVE ZERO TO WO3-OPTIONS.
           COMPUTE WO3-OPTIONS    =  MQOO-INPUT-SHARED.
           DISPLAY 'OPEN OPTIONS' WO3-OPTIONS.
      *
           CALL 'MQOPEN' USING WO3-HCONN
                               MQOD
                               WO3-OPTIONS
                               WO3-HOBJ-CLIENTQ
                               WO3-COMPCODE
                               WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'         TO  MO1-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
              MOVE WO3-COMPCODE     TO  MO1-MSG4-COMPCODE
              MOVE WO3-REASON       TO  MO1-MSG4-REASON
              MOVE MO1-MESSAGE-4    TO  WOO-MESSAGE
              MOVE WOO-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO TO A-MAIN-EXIT
           END-IF.
      *
      *
       GET-CLIENT-PARM-NEXT.
           COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                                   MQGMO-NO-SYNCPOINT.
      *
      *
      *  WAIT UNLIMITED FOR REQUESTS FROM WEBCLIENT
      *
      *
           MOVE MQWI-UNLIMITED     TO MQGMO-WAITINTERVAL.
```

Figure 38 (Part 35 of 40). MVS WTPING Cobol Program Source

```
      *
           MOVE SPACE TO WOO-PARM-BUFFER.
           MOVE LENGTH OF WOO-PARM-BUFFER TO WO3-BUFFLEN.
           MOVE MQMI-NONE               TO MQMD-MSGID.
           MOVE MQCI-NONE               TO MQMD-CORRELID.
      *
      *    MAKE THE MQGET CALL TO CLIENT INPUT QUEUE
      *
           CALL 'MQGET' USING WO3-HCONN
                              WO3-HOBJ-CLIENTQ
                              MQMD
                              MQGMO
                              WO3-BUFFLEN
                              WOO-PARM-BUFFER
                              WO3-DATALEN
                              WO3-COMPCODE
                              WO3-REASON.
      *

            IF WO3-COMPCODE = MQCC-FAILED
              MOVE 'MQGET'          TO  MO1-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  MO1-MSG4-OBJECTNAME
              MOVE WO3-COMPCODE    TO  MO1-MSG4-COMPCODE
              MOVE WO3-REASON      TO  MO1-MSG4-REASON
              MOVE MO1-MESSAGE-4   TO  WOO-MESSAGE
              MOVE WOO-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO TO A-MAIN-EXIT
            END-IF.
      *
      *
      *
      * STRIP OFF HEADER(47) AND NL-CHARACTER
      *
           COMPUTE WO3-DATALEN-VALID = WO3-DATALEN - 48.
      *
           MOVE MQMD-MSGID     TO  WO3-CLIENT-MSGID
           MOVE MQMD-CORRELID  TO  WO3-CLIENT-CORRELID
           MOVE MQMD-REPLYTOQ  TO  WO3-CLIENT-RESP-QNAME
           MOVE MQMD-REPLYTOQMGR TO WO3-CLIENT-RESP-QMGR
```

Figure 38 (Part 36 of 40). MVS WTPING Cobol Program Source

```
           PERFORM BUILD-PARM.
      *
      *
      *
           CALL 'MQCLOSE' USING W03-HCONN
                                W03-HOBJ-CLIENTQ
                                MQCO-NONE
                                W03-COMPCODE
                                W03-REASON.
      *
           IF W03-COMPCODE NOT = MQCC-OK
              MOVE 'MQCLOSE'       TO  M01-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  M01-MSG4-OBJECTNAME
              MOVE W03-COMPCODE    TO  M01-MSG4-COMPCODE
              MOVE W03-REASON      TO  M01-MSG4-REASON
              MOVE M01-MESSAGE-4   TO  W00-MESSAGE
              MOVE W00-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO   TO A-MAIN-EXIT
           END-IF.
      *
      *
       GET-CLIENT-PARMS-EXIT.
           EXIT.
           EJECT
       PUT-MSG-TO-CLIENT SECTION.
      * ------------------------------------------------------------- *
      *    SEND MESSAGE BACK TO CLIENT
      * ------------------------------------------------------------- *
           MOVE MQOT-Q                  TO  MQOD-OBJECTTYPE.
           MOVE W03-CLIENT-RESP-QNAME   TO  MQOD-OBJECTNAME.
           MOVE W03-CLIENT-RESP-QMGR    TO  MQOD-OBJECTQMGRNAME.

      *
           COMPUTE W03-OPTIONS = MQOO-OUTPUT
      *
           MOVE ZERO  TO  W03-HOBJ-CLIENT-RESPQ
      *
           CALL 'MQOPEN'  USING W03-HCONN
                                MQOD
                                W03-OPTIONS
                                W03-HOBJ-CLIENT-RESPQ
                                W03-COMPCODE
                                W03-REASON.
```

Figure 38 (Part 37 of 40). MVS WTPING Cobol Program Source

```
      *
           IF W03-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'         TO  M01-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  M01-MSG4-OBJECTNAME
              MOVE W03-COMPCODE    TO  M01-MSG4-COMPCODE
              MOVE W03-REASON      TO  M01-MSG4-REASON
              MOVE M01-MESSAGE-4   TO  W00-MESSAGE
              MOVE W00-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO   TO A-MAIN-EXIT
           END-IF.
              COMPUTE MQPMO-OPTIONS = MQPMO-NO-SYNCPOINT.
              MOVE W03-CLIENT-MSGID    TO MQMD-CORRELID.
              MOVE MQMI-NONE           TO MQMD-MSGID.
      *
              MOVE LENGTH OF W03-REPLY-CLIENT-MSG TO W03-DATALEN.
      *
      *
      * SPECIFY THE FORMAT AS STRING
      * MQGate expects this!
      *
      *
              MOVE 273          TO MQMD-ENCODING.
              MOVE 850          TO MQMD-CODEDCHARSETID.
      *
      *     EBCDIC TO ASCII
      *
              MOVE SPACE TO TRAN-BUFFER.
              MOVE LENGTH OF W03-REPLY-CLIENT-MSG2 TO TRAN-BUF-LEN.
              MOVE W03-REPLY-CLIENT-MSG2 TO TRAN-BUFFER.
              PERFORM TRAN-MSG.
              MOVE TRAN-BUFFER TO W03-REPLY-CLIENT-MSG2.
      *
      *
              CALL 'MQPUT'  USING  W03-HCONN
                                   W03-HOBJ-CLIENT-RESPQ
                                   MQMD
                                   MQPMO
                                   W03-DATALEN
                                   W03-REPLY-CLIENT-MSG
                                   W03-COMPCODE
                                   W03-REASON.
```

Figure 38 (Part 38 of 40). MVS WTPING Cobol Program Source

```
           IF W03-COMPCODE NOT = MQCC-OK
              MOVE 'MQPUT '        TO  M01-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  M01-MSG4-OBJECTNAME
              MOVE W03-COMPCODE    TO  M01-MSG4-COMPCODE
              MOVE W03-REASON      TO  M01-MSG4-REASON
              MOVE M01-MESSAGE-4   TO  W00-MESSAGE
              MOVE W00-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO   TO A-MAIN-EXIT
           END-IF.
           MOVE SPACE TO W03-REPLY-CLIENT-MSG.
      *
      *    CLOSE REPLY QUEUE TO CLIENT
      *
           CALL 'MQCLOSE' USING W03-HCONN
                                W03-HOBJ-CLIENT-RESPQ
                                MQCO-NONE
                                W03-COMPCODE
                                W03-REASON.
      *
           IF W03-COMPCODE NOT = MQCC-OK
              MOVE 'MQCLOSE'       TO  M01-MSG4-OPERATION
              MOVE MQOD-OBJECTNAME TO  M01-MSG4-OBJECTNAME
              MOVE W03-COMPCODE    TO  M01-MSG4-COMPCODE
              MOVE W03-REASON      TO  M01-MSG4-REASON
              MOVE M01-MESSAGE-4   TO  W00-MESSAGE
              MOVE W00-MESSAGE TO OUTMSG
              PERFORM DISPLAY-CONS
              GO   TO A-MAIN-EXIT
           END-IF.
      *
      *
      *
       PUT-MSG-TO-CLIENT-EXIT.
           EXIT.
       BUILD-PARM  SECTION.
      * ------------------------------------------------------------- *
      *
           MOVE W00-PARM-STRING TO W00-PARM-ARRAY.
           DISPLAY W03-DATALEN-VALID.
```

Figure 38 (Part 39 of 40). MVS WTPING Cobol Program Source

```
           DISPLAY WOO-PARM-ARRAY.
           UNSTRING WOO-PARM-ARRAY DELIMITED BY ALL '&'
                                   INTO WO3-NOMSG
                                        WO3-LENMSG
                                        WO3-FBYTES
                                        WO3-REMQMGR
                                        WO3-COMMENT
                                        WO3-RESPMSG.
           DISPLAY WO3-NOMSG.
           DISPLAY WO3-LENMSG.
           DISPLAY WO3-FBYTES.
           DISPLAY WO3-REMQMGR.
           DISPLAY WO3-COMMENT
           DISPLAY WO3-RESPMSG
           MOVE    WO3-NOMSG-PARM TO WOO-MSG-TOTAL.
           DISPLAY 'PARMS FROM CLIENT Q'.
           DISPLAY WOO-MSG-TOTAL.
           MOVE    WO3-LENMSG-PARM TO WOO-MSG-LEN.
           DISPLAY WOO-MSG-LEN.
           MOVE    WO3-FBYTES-PARM TO WOO-FIRSTTXT.
           DISPLAY WOO-FIRSTTXT.
           MOVE    WO3-REMQMGR-PARM TO WO3-TGTQMGR.
           DISPLAY WO3-TGTQMGR.
           MOVE    WO3-COMMENT-PARM TO WOO-COMMENT.
           DISPLAY WOO-COMMENT.
           MOVE    WO3-RESPMSG-PARM TO WOO-MSG-GETRSP.
           DISPLAY WOO-MSG-GETRSP.
      *
       BUILD-PARM-EXIT.
           EXIT.
       TRAN-MSG  SECTION.
      * ------------------------------------------------------------ *
      *
      *   PERFORM THE RELEVANT CONVERSION
      *   NOTE THAT IF WE DON'T FIND A MATCH ON ANY CHARACTER,
      *   WE LEAVE THE DATA ASIS
      *
      *   CHANGE 500 TO 850 CCSID
      *
           PERFORM VARYING WS-SUB FROM 1   BY 1
                   UNTIL WS-SUB > TRAN-BUF-LEN
                      SET WS-EBCDIC-INDX TO 1
                      SEARCH LS-EBCDIC-ENTRY
                        WHEN LS-EBCDIC-ENTRY(WS-EBCDIC-INDX)
                          = TRAN-ARRAY(WS-SUB)
                            MOVE LS-ASCII-ENTRY(WS-EBCDIC-INDX)
                             TO TRAN-ARRAY(WS-SUB)
                      END-SEARCH
           END-PERFORM.
       TRAN-MSG-EXIT.
           EXIT.
```

Figure 38 (Part 40 of 40). MVS WTPING Cobol Program Source

C.3 MVS WTPONG Cobol Source

```
CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
      * ------------------------------------------------------------- *
       IDENTIFICATION DIVISION.
      * ------------------------------------------------------------- *
       PROGRAM-ID. PRGQJPS8.
      *REMARKS
      * ------------------------------------------------------------- *
      * Server program for subarea SA25 + SA18
      *
      *
      *
      *
      * Batch server :
      *
      * This server runs as a batch job (jcl runjps8)
      * It stays active until a termination message arrives
      * or the job is canceled by an operator.
      * Batch job accepts a parm card like PARM='QMGR,REQ.QUEUE'
      * This program is a a batch clone from the corresponding
      * CICS server transaction program PRGQMIS3.
      *
      *
      *
      * Name of the program: PRGQJ P S   8:
      *                      !     ! !   !
      *                      !     ! !   +---> team name (now fixed)
      *                      !     ! +-------> Server program
      *                      !     !
      *                      !     +---------> S.A. 25
      *                      +---------------> Fixed identifier
      *
      * Input  (request) queue: xxx.CSQ1.REQUESTQ (fixed)
      * Output (reply)   queue: comes from ReplyToQueue field
      *
      * NOTE: when cloning this program, mind the queue manager name:
      *       they must be changed from CSQ1 to something else
      ****************************************************************
      *   @START_COPYRIGHT@                                          *
      *  Environment           : batch                               *
      *                                                              *
```

Figure 39 (Part 1 of 22). MVS WTPONG Cobol Program Source

```
*  Function    : This program processes messages from the       *
*                request queue and put responses to the         *
*                response queue                                 *
*                                                               *
* ************************************************************* *
   EJECT
* ************************************************************* *
*                                                               *
*                   Program logic                               *
*                   -------------                               *
*                                                               *
*START.                                                         *
*    open request queue.                                        *
*    if open fails                                              *
*       write error to log                                      *
*       exit program                                            *
*                                                               *
*    Get first msg from the request queue.                      *
*    if get fails                                               *
*       exit the program                                        *
*    Get replytoqueue from message descriptor                   *
*                                                               *
*    Perform until compcode not = ok                            *
*            Evaluate msg-type                                  *
*                when 1st-msg                                   *
*                     perform ask-datetime                      *
*                     open replytoqueue (exit if error)         *
*                     perform put-respq                         *
*                     close replytoqueue (exit if error)        *
*                when intm-msg                                  *
*                     perform ask-datetime                      *
*                when last-msg                                  *
*                     perform ask-datetime                      *
*                     open replytoqueue (exit if error)         *
*                     perform put-respq                         *
*                     close replytoqueue (exit if error)        *
*            End-evaluate                                       *
*            get next msg from request-queue                    *
*    end-perform.                                               *
*    if unexpected compcode                                     *
```

Figure 39 (Part 2 of 22). MVS WTPONG Cobol Program Source

```
*        exit program with error message                        *
*    if no-msg-available                                        *
*       exit program w/o any error message                      *
*    close all queues.                                          *
*                                                               *
* ************************************************************* *
 EJECT
* ------------------------------------------------------------- *
 ENVIRONMENT DIVISION.
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 DATA DIVISION.
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 WORKING-STORAGE SECTION.
* ------------------------------------------------------------- *
*
*    W00 - General work fields
*
 01  M01-MESSAGE-CON.
     03  M01-CON1               PIC X(22).
     03  M01-CON2               PIC X(50).
 01  M02-MESSAGE-CON.
     03  M02-CON1               PIC X(22).
     03  M02-CON2               PIC 9(05).
 01  M03-MESSAGE-CON.
     03  M03-CON1               PIC X(22).
     03  M03-CON2               PIC 9(05).
 01  W00-MESSAGE-COUNT          PIC S9(9) BINARY.
 01  W00-MESSAGE                PIC X(70).
 01  W00-WAIT-INTERVAL          PIC S9(09) BINARY VALUE 10.
 01  W00-INPUT-MSG-PRIORITY     PIC S9(09) BINARY.
 01  W00-SUB                    PIC S9(09) BINARY.
 01  W00-INDEX                  PIC S9(09) BINARY.
*
*    Conn parms
*
 01  W02-MQM                    PIC X(48) VALUE 'CSQ1'.
 01  W02-DEBUG                  PIC X(7)  VALUE SPACES.
*
```

Figure 39 (Part 3 of 22). MVS WTPONG Cobol Program Source

```
      *    Queue names
      *
       01  WO2-QUEUE-NAMES.
           05  WO2-REQUEST-QNAME       PIC X(48) VALUE
           'DXB.CSQ1.REQUESTQ                          '.
           05  WO2-RESPONSE-QMGR       PIC X(48) VALUE SPACES.
           05  WO2-RESPONSE-QNAME      PIC X(48) VALUE
           'DUMMY                                      '.
      *
      *    WO3 - MQM API fields
      *
       01  WO3-SELECTORCOUNT           PIC S9(9) BINARY VALUE 1.
       01  WO3-INTATTRCOUNT            PIC S9(9) BINARY VALUE 1.
       01  WO3-CHARATTRLENGTH          PIC S9(9) BINARY VALUE ZERO.
       01  WO3-CHARATTRS               PIC X     VALUE LOW-VALUES.
       01  WO3-HCONN                   PIC S9(9) BINARY VALUE ZERO.
       01  WO3-OPTIONS                 PIC S9(9) BINARY.
       01  WO3-HOBJ-REQUESTQ           PIC S9(9) BINARY.
       01  WO3-HOBJ-RESPQ              PIC S9(9) BINARY.
       01  WO3-HOBJ-DEADQ              PIC S9(9) BINARY.
       01  WO3-COMPCODE                PIC S9(9) BINARY.
       01  WO3-REASON                  PIC S9(9) BINARY.
       01  WO3-TERMINATE               PIC X(65) VALUE 'TERM'.
       01  WO3-SELECTORS-TABLE.
           05  WO3-SELECTORS           PIC S9(9) BINARY OCCURS 2 TIMES.
       01  WO3-INTATTRS-TABLE.
           05  WO3-INTATTRS            PIC S9(9) BINARY OCCURS 2 TIMES.
       01  WO3-DATALEN                 PIC S9(9) BINARY.
       01  WO3-BUFFLEN                 PIC S9(9) BINARY.
       01  GET-COMPCODE                PIC S9(9) BINARY.
       01  GET-REASON                  PIC S9(9) BINARY.
      *
       01  WO3-PUT-BUFFER.
           COPY PRGQO1X2.
      **   03  WO3-RESPONSE.
      **       05  WO3-RESP-MSGID          PIC X(8).
      **       05  WO3-RESP-CORRELID       PIC X(24).
      **       05  WO3-RESP-GROUP.
      **           10  WO3-RESP-MSG-TOT    PIC 9(3).
      **           10  WO3-RESP-MSG-LEN    PIC 9(6).
```

Figure 39 (Part 4 of 22). MVS WTPONG Cobol Program Source

```
**          10  WO3-RESP-SEND-TIME  PIC X(16).
**          10  WO3-RESP-RECV-TIME  PIC X(16).
**          10  WO3-RESP-RESP-TIME  PIC X(16).
*
 01  WO3-GET-BUFFER.
     COPY PRGQO1X1.
     03      REDEFINES WO3-REQ-MSG.
         05  WO3-REQ-MSG-TXT-5OBYTE  PIC X(50).
         05  FILLER                  PIC X(9950).
*
**   03  WO3-REQUEST.
**       05  WO3-REQ-MSGID           PIC X(8).
**           88 WO3-REQ-FIRST-MSG    VALUE 'FIRST   '.
**           88 WO3-REQ-INTERMED-MSG VALUE 'INTERMED'.
**           88 WO3-REQ-LAST-MSG     VALUE 'LAST    '.
**       05  WO3-REQ-CORRELID        PIC X(24).
**       05  WO3-REQ-GROUP.
**           10  WO3-REQ-MSG-TOT     PIC 9(3).
**           10  WO3-REQ-MSG-LEN     PIC 9(6).
**           10  WO3-REQ-SEND-TIME   PIC X(16).
**           10  WO3-REQ-RECV-TIME   PIC X(16).
**           10  WO3-REQ-RESP-TIME   PIC X(16).
**   03  WO3-REQ-MSG.
**       05  WO3-REQ-MSG-SEG         PIC X(10)  OCCURS 1000.
*
 01  WO3-REQUEST-HDR.
     03  WO3-REQ-MSGID           PIC X(8).
         88 WO3-REQ-FIRST-MSG    VALUE 'FIRST   '.
         88 WO3-REQ-INTERMED-MSG VALUE 'NOTFIRST'.
         88 WO3-REQ-LAST-MSG     VALUE 'LAST    '.
     03  WO3-REQ-CORRELID        PIC X(24).
*
 01  FIRST-ASCII         PIC X(5) VALUE X'4649525354'.
 01  NOT-FIRST-ASCII     PIC X(8) VALUE X'4E4F544649525354'.
 01  LAST-ASCII          PIC X(4) VALUE X'4C415354'.
*
*    Message Output-Buffer
*
 01  OUTMSG                      PIC X(120) VALUE SPACE.
*
```

Figure 39 (Part 5 of 22). MVS WTPONG Cobol Program Source

```
      *    diagnostic trace
      *
       01  DIAG-MSG                    PIC X(80).
      *
      *    message format (Request/Response message)
      *
       01  WO3-TS-REQ-RSP.
           03  WO3-TS-MSG-HDR.
               05  WO3-TS-MSGID        PIC X(8).
               05  WO3-TS-CORRELID     PIC X(24).
           03                           PIC X(01).
           03  WO3-TS-MSG-TOT          PIC X(5).
           03                           PIC X(01).
           03  WO3-TS-MSG-LEN          PIC X(6).
           03                           PIC X(01).
           03  WO3-TS-REQ-PUTTIME      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-REQ-PUTDATE      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-REQ-GETTIME      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-REQ-GETDATE      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-RSP-PUTTIME      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-RSP-PUTDATE      PIC X(10).
           03                           PIC X(01).
           03  WO3-TS-MSG-TXT-50BYTE   PIC X(50).
           03                           PIC X(01).
           03  WO3-TS-MSG-COMMENT      PIC X(65).
      *
      *
      *    API control blocks
      *
       01  MQM-OBJECT-DESCRIPTOR.
           COPY CMQODV.
       01  MQM-MESSAGE-DESCRIPTOR.
           COPY CMQMDV.
       01  MQM-PUT-MESSAGE-OPTIONS.
           COPY CMQPMOV.
```

Figure 39 (Part 6 of 22). MVS WTPONG Cobol Program Source

```
 01  MQM-GET-MESSAGE-OPTIONS.
     COPY CMQGMOV.
 01  MQM-TRIGGER-MESSAGE.
     COPY CMQTML.
*
*
 01  WO5-DATESEP              PIC X VALUE '-'.
 01  WO5-TIMESEP              PIC X VALUE ':'.
 01  WO5-TIMENEW              PIC X(8).
 01  WO5-DATENEW              PIC X(8).
 01  WO5-DATETIME.
     03  WO5-DATE.
        05 WO5-YY             PIC X(2).
        05 WO5-MM             PIC X(2).
        05 WO5-DD             PIC X(2).
        05 FILLER             PIC X(2).
     03  WO5-TIME.
        05  WO5-HH               PIC X(2).
        05  WO5-MI               PIC X(2).
        05  WO5-SS               PIC X(2).
        05  WO5-TT               PIC X(2).
*
*    PRGQO1X9 contains error message to be written to diag/cons
*
 COPY PRGQO1X9.
*
*    main process flags
*
 01  WO6-MAIN-PROCESS-FLAG    PIC 9 VALUE 0.
     88 END-PROCESS VALUE 1.
 01  WO6-END-PROCESS          PIC 9 VALUE 1.
*
 01  WO6-INQUIRYQ-STATUS      PIC X(6) VALUE 'CLOSED'.
     88 INQUIRYQ-OPEN   VALUE 'OPEN'.
     88 INQUIRYQ-CLOSED VALUE 'CLOSED'.
*
 01  WO6-CALL-STATUS          PIC X(6) VALUE 'OK'.
     88 CALLS-OK       VALUE 'OK'.
 01  WO6-CALL-ERROR           PIC X(6) VALUE 'FAILED'.
*
```

Figure 39 (Part 7 of 22). MVS WTPONG Cobol Program Source

```
 01  WO6-MSG-STATUS           PIC 9 VALUE 0.
     88 MSG-COMPLETE     VALUE 1.
     88 MSG-NOT-COMPLETE VALUE 0.
*
*
*    MQV contains constants (for filling in the control blocks)
*    and return codes (for testing the result of a call)
*
 01  W99-MQV.
 COPY CMQV SUPPRESS.
 LINKAGE SECTION.
* ------------------------------------------------------------- *
 01  PARMDATA.
     05  PARM-LEN               PIC S9(03) BINARY.
     05  PARM-STRING            PIC X(100).
*
* ------------------------------------------------------------- *
 PROCEDURE DIVISION USING PARMDATA.
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 A-MAIN SECTION.
* ------------------------------------------------------------- *
*
* This section interprets the parm card
* and selects queue-manager and request-queue from jcl-parm
*
* ------------------------------------------------------------- *
 PARM-INFO-SELECT.
*
*    If no data was passed, create a message, print it, and
*    exit
*
     IF PARM-LEN = 0 THEN
        STRING 'MQS95001 : 95001 SA25 CSQ1 '
               'NO PARM-CARD ACCEPTING DEFAULT '
               'CSQ1 AND DXB-CSQ1-REQUESTQ'
                DELIMITED BY SIZE INTO OUTMSG
        PERFORM DISPLAY-CONS
        GO TO APPL-CONNECT-QMGR
     END-IF.
```

Figure 39 (Part 8 of 22). MVS WTPONG Cobol Program Source

```
*
*    Separate into the relevant fields any data passed in the
*    PARM statement
*
     UNSTRING PARM-STRING DELIMITED BY ALL ',,'
                               INTO WO2-MQM
                                    WO2-REQUEST-QNAME
                                    WO2-DEBUG.
*
*    check for valid queuemanager and queue combination
*
     IF WO2-MQM = 'CSQ1' THEN GO TO APPL-CONNECT-QMGR.
     IF WO2-MQM = 'CSQ2' THEN GO TO APPL-CONNECT-QMGR.
     STRING 'MQS95002 : 95002          '
            'No valid queuemanager in PARM '
            DELIMITED BY SIZE INTO OUTMSG
       PERFORM DISPLAY-CONS
       GO   TO A-MAIN-EXIT.
 APPL-CONNECT-QMGR.
*    check for queue name
     IF WO2-REQUEST-QNAME = SPACE THEN
     STRING 'MQS95003 : 95003          '
            'No requestq in PARM '
            DELIMITED BY SIZE INTO OUTMSG
       PERFORM DISPLAY-CONS
       GO   TO A-MAIN-EXIT
     END-IF.
* ------------------------------------------------------------ *
*                                                              *
* This section connects the server to Qmgr                     *
*                                                              *
* ------------------------------------------------------------ *
*
*
*    Connect to the specified queue manager.
*
     CALL 'MQCONN' USING WO2-MQM
                         WO3-HCONN
                         WO3-COMPCODE
                         WO3-REASON.
```

Figure 39 (Part 9 of 22). MVS WTPONG Cobol Program Source

```
      *
      *    Test the output of the connect call.  If the call failed,
      *    print an error message showing the completion code and
      *    reason code
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              PERFORM RECORD-CALL-ERROR
              GO   TO A-MAIN-EXIT
           END-IF
           IF WO2-MQM = 'CSQ1' then
             STRING 'MQS95001 : 95001 SA25 CSQ1 '
                       'Server connected '
                      DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
            else
             STRING 'MQS95001 : 95001 SA18 CSQ2 '
                       'Server connected '
                      DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
            end-if.
       OPEN-REQUEST-Q.
      * ------------------------------------------------------------ *
      *                                                              *
      * This section opens the request queue for input shared        *
      *                                                              *
      * ------------------------------------------------------------ *
      *
           MOVE MQOT-Q            TO MQOD-OBJECTTYPE.
           MOVE WO2-REQUEST-QNAME TO MQOD-OBJECTNAME.
      *
           COMPUTE WO3-OPTIONS    =  MQOO-INPUT-SHARED +
                                     MQOO-SAVE-ALL-CONTEXT.
      *
           CALL 'MQOPEN' USING WO3-HCONN
                               MQOD
                               WO3-OPTIONS
                               WO3-HOBJ-REQUESTQ
                               WO3-COMPCODE
                               WO3-REASON.
      *
```

Figure 39 (Part 10 of 22). MVS WTPONG Cobol Program Source

```
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQOPEN'             TO MO2-OPERATION
              MOVE WO2-REQUEST-QNAME    TO MO2-OBJECTNAME
              PERFORM RECORD-CALL-ERROR
              GO   TO A-MAIN-EXIT
           END-IF.
      *
      *
       MAIN-PROCESS.
      *-------------------------------------------------------------- *
      *    Loop until messages are in the queue
      *-------------------------------------------------------------- *
      *
      *    Initialize the Get Message Options (MQGMO) control block.
      *    (The copy book initializes the remaining fields)
      *
           IF WO2-MQM = 'CSQ1' then
             STRING 'MQS95001 : 9501 SA25 CSQ1 '
                       'Server listening on '  WO2-REQUEST-QNAME
                    DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
            else
             STRING 'MQS95001 : 9501 SA18 CSQ2 '
                       'Server listening on '  WO2-REQUEST-QNAME
                    DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
            end-if.
           COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                                   MQGMO-ACCEPT-TRUNCATED-MSG +
                                   MQGMO-NO-SYNCPOINT.
      *
      *
      *  wait unlimited as a server
      *
      *
           MOVE MQWI-UNLIMITED     TO MQGMO-WAITINTERVAL
      *
           MOVE LENGTH OF WO3-GET-BUFFER TO WO3-BUFFLEN.
           MOVE MQMI-NONE                TO MQMD-MSGID.
           MOVE MQCI-NONE                TO MQMD-CORRELID.
```

Figure 39 (Part 11 of 22). MVS WTPONG Cobol Program Source

```
      *
      *    Make the first MQGET call outside the loop
      *
           CALL 'MQGET' USING W03-HCONN
                              W03-HOBJ-REQUESTQ
                              MQMD
                              MQGMO
                              W03-BUFFLEN
                              W03-GET-BUFFER
                              W03-DATALEN
                              GET-COMPCODE
                              GET-REASON.
      *
      *
            IF GET-COMPCODE = MQCC-FAILED
              MOVE 'MQGET FIRST '    TO M02-OPERATION
              MOVE W02-REQUEST-QNAME TO M02-OBJECTNAME
              MOVE GET-COMPCODE  TO W03-COMPCODE
              MOVE GET-REASON    TO W03-REASON
              PERFORM RECORD-CALL-ERROR
              GO TO A-MAIN-EXIT
            END-IF
      *
      *  write out environment parms to sysout dd
      *
      *
            MOVE '+SERV GET FIRST=' TO M01-CON1
            MOVE M01-CON1 TO OUTMSG
            PERFORM DISPLAY-LOCAL
            MOVE W03-REQ-MSG-TXT-50BYTE       TO M01-CON2
            MOVE M01-CON2 TO OUTMSG
            PERFORM DISPLAY-LOCAL
            MOVE '+SERV CCSID    =' TO M02-CON1
            MOVE M02-CON1 TO OUTMSG
            PERFORM DISPLAY-LOCAL
            MOVE MQMD-CODEDCHARSETID          TO M02-CON2
            MOVE M02-CON2 TO OUTMSG
            PERFORM DISPLAY-LOCAL
            MOVE '+SERV LENGTH   =' TO M03-CON1
            MOVE M03-CON1 TO OUTMSG
```

Figure 39 (Part 12 of 22). MVS WTPONG Cobol Program Source

```
           PERFORM DISPLAY-LOCAL
           MOVE WO3-DATALEN                  TO MO3-CON2
           MOVE MO3-CON2 TO OUTMSG
           PERFORM DISPLAY-LOCAL
      *
           MOVE MQMD-MSGID     TO  WO3-REQ-MSGID
           MOVE MQMD-CORRELID  TO  WO3-REQ-CORRELID
      *
      *    Loop from here to END-PERFORM until the MQGET call fails
      *
           PERFORM WITH TEST BEFORE
                   UNTIL GET-COMPCODE = MQCC-FAILED
      *
      *
      *
      *    Server handles Termination Message
      *    If message commet = TERM then process all
      *    messages on the message queue and terminate
      *    the server: switch unlimited waitinterval
      *    to 10 sec-interval
      *
      *
             IF WO3-REQ-MSG-COMMENT = WO3-TERMINATE then
               MOVE WOO-WAIT-INTERVAL TO MQGMO-WAITINTERVAL
               IF WO2-MQM = 'CSQ1' then
                   STRING 'MQS95OO1 : 95OO1 SA25 CSQ1 '
                          'Server received terminate message'
                       DELIMITED BY SIZE INTO OUTMSG
                   PERFORM DISPLAY-CONS
                ELSE
                   STRING 'MQS95OO1 : 95OO1 SA18 CSQ2 '
                          'Server received terminate message'
                       DELIMITED BY SIZE INTO OUTMSG
                   PERFORM DISPLAY-CONS
               END-IF
             END-IF
      *
      *
      *
```

Figure 39 (Part 13 of 22). MVS WTPONG Cobol Program Source

```
      *
      *       Save ReplyToQueue name
      *
      *
              MOVE SPACES         TO  WO3-TS-REQ-RSP
      *
      *
      *       DISPLAY NO OF MESSAGES PROCESSED PER BATCH
      *
      *
      *
              IF WO3-REQ-FIRST-MSG OR WO3-REQ-MSGID(1:5) = FIRST-ASCII
                   MOVE 1 TO WOO-MESSAGE-COUNT
                 ELSE
                   ADD 1 TO WOO-MESSAGE-COUNT
              END-IF
              IF WO3-REQ-LAST-MSG OR WO3-REQ-MSGID(1:4) = LAST-ASCII
                 THEN
                DISPLAY 'NO. OF MESSAGES PROCESSED ' WOO-MESSAGE-COUNT
              END-IF
      *
      *
              EVALUATE TRUE
                  WHEN WO3-REQ-FIRST-MSG  OR  WO3-REQ-LAST-MSG OR
                       WO3-REQ-MSGID(1:5) = FIRST-ASCII      OR
                       WO3-REQ-MSGID(1:4) = LAST-ASCII
                      MOVE MQMD-REPLYTOQ TO WO2-RESPONSE-QNAME
                      MOVE MQMD-REPLYTOQMGR TO WO2-RESPONSE-QMGR
                      PERFORM ASK-DATETIME
                      PERFORM MOVE-REQ-TO-TS
                      PERFORM OPEN-RESPONSE-Q
                      PERFORM PUT-RESPONSE-MESSAGE
                      PERFORM CLOSE-RESPONSE-Q
                      PERFORM MOVE-RSP-TO-TS
                      PERFORM FORWARD-MSG-TO-TSQ
      *
                  WHEN WO3-REQ-INTERMED-MSG OR
                       WO3-REQ-MSGID(1:8) = NOT-FIRST-ASCII
                      PERFORM ASK-DATETIME
                      PERFORM MOVE-REQ-TO-TS
```

Figure 39 (Part 14 of 22). MVS WTPONG Cobol Program Source

```
                    PERFORM FORWARD-MSG-TO-TSQ
      *
                WHEN OTHER
                    MOVE 'UNKNOWN MSGID'    TO MO2-OPER-MSG
                    MOVE MQMD-MSGID         TO MO2-OPER-MSGID
                    MOVE MQMD-CORRELID      TO MO2-OPER-CORRID
                    MOVE WO2-REQUEST-QNAME TO MO2-OBJECTNAME
                    PERFORM RECORD-CALL-ERROR
            END-EVALUATE
      *
      *       Clear MQMD-MSGID and MQMD-CORRELID before the next
      *       MQGET call to ensure that all messages are retrieved
      *
            MOVE MQMI-NONE TO MQMD-MSGID
            MOVE MQCI-NONE TO MQMD-CORRELID
      *
      *       Get the next message
      *
            CALL 'MQGET' USING WO3-HCONN
                               WO3-HOBJ-REQUESTQ
                               MQMD
                               MQGMO
                               WO3-BUFFLEN
                               WO3-GET-BUFFER
                               WO3-DATALEN
                               GET-COMPCODE
                               GET-REASON
      *
            MOVE MQMD-MSGID     TO  WO3-REQ-MSGID
            MOVE MQMD-CORRELID  TO  WO3-REQ-CORRELID
      *
      *       Test the output of the MQGET call at the top of the loop.
      *       Exit the loop if an error occurs
      *
          END-PERFORM.
      *
      *    Test the output of the MQGET call.  If the call failed,
      *    print an error message showing the completion code and
      *    reason code, unless the reason code is NO-MSG-AVAILABLE.
      *
```

Figure 39 (Part 15 of 22). MVS WTPONG Cobol Program Source

```
*    Note: When the loop reaches the end of the file, the
*          completion code is MQCC-FAILED and the reason code
*          is MQRC-NO-MSG-AVAILABLE. If this happens, the
*          program closes with no error message
*
     IF GET-REASON   NOT = MQRC-NO-MSG-AVAILABLE
        MOVE 'MQGET NEXT  '     TO MO2-OPERATION
        MOVE WO2-REQUEST-QNAME TO MO2-OBJECTNAME
        MOVE GET-COMPCODE      TO WO3-COMPCODE
        MOVE GET-REASON        TO WO3-REASON
        PERFORM RECORD-CALL-ERROR
     END-IF.
*
*
 CLOSE-ALL.
*
     PERFORM CLOSE-REQUEST-Q.
*
 A-MAIN-EXIT.
*
*
*
     GOBACK.
     EJECT
*
 OPEN-RESPONSE-Q SECTION.
* ------------------------------------------------------------ *
*                                                              *
* This section opens the response queue for output             *
* The response queue name to open is in WO2-RESPONSE-QNAME     *
*                                                              *
* ------------------------------------------------------------ *
*
     MOVE MQOT-Q             TO MQOD-OBJECTTYPE.
     MOVE WO2-RESPONSE-QNAME TO MQOD-OBJECTNAME.
     MOVE WO2-RESPONSE-QMGR  TO MQOD-OBJECTQMGRNAME.
*
     COMPUTE WO3-OPTIONS    =  MQOO-OUTPUT.
*
     CALL 'MQOPEN' USING WO3-HCONN
                         MQOD
                         WO3-OPTIONS
                         WO3-HOBJ-RESPQ
                         WO3-COMPCODE
                         WO3-REASON.
```

Figure 39 (Part 16 of 22). MVS WTPONG Cobol Program Source

```
      *
           IF WO3-COMPCODE NOT = MQCC-OK
               MOVE 'MQOPEN'              TO MO2-OPERATION
               MOVE WO2-RESPONSE-QNAME    TO MO2-OBJECTNAME
               PERFORM RECORD-CALL-ERROR
               GO   TO A-MAIN-EXIT
           END-IF.
      *
       OPEN-RESPONSE-Q-EXIT.
           EXIT.
           EJECT
      *
      * ------------------------------------------------------------ *
       CLOSE-REQUEST-Q SECTION.
      * ------------------------------------------------------------ *
      *                                                              *
      * This section closes the request queue                        *
      *                                                              *
      * ------------------------------------------------------------ *
      *
           CALL 'MQCLOSE' USING WO3-HCONN
                                WO3-HOBJ-REQUESTQ
                                MQCO-NONE
                                WO3-COMPCODE
                                WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
               MOVE 'MQCLOSE'             TO MO2-OPERATION
               MOVE WO2-REQUEST-QNAME     TO MO2-OBJECTNAME
               PERFORM RECORD-CALL-ERROR
               GO TO A-MAIN-EXIT
           END-IF.
      *
      * Program sends a close message to the system log
      *
           IF WO2-MQM = 'CSQ1' then
             STRING 'MQS95001 : 95001 SA25 CSQ1 '
                       'Server ended '
                    DELIMITED BY SIZE INTO OUTMSG
             PERFORM DISPLAY-CONS
```

Figure 39 (Part 17 of 22). MVS WTPONG Cobol Program Source

```
           else
            STRING 'MQS95001 : 95001 SA18 CSQ2 '
                     'Server ended '
                    DELIMITED BY SIZE INTO OUTMSG
            PERFORM DISPLAY-CONS
           end-if.
      *
       CLOSE-REQUEST-Q-EXIT.
      *
      *    Return to performing section
      *
           EXIT.
           EJECT
      *
      * ---------------------------------------------------------- *
       CLOSE-RESPONSE-Q SECTION.
      * ---------------------------------------------------------- *
      *                                                            *
      * This section closes the response queue                     *
      *                                                            *
      * ---------------------------------------------------------- *
      *
           CALL 'MQCLOSE' USING W03-HCONN
                                W03-HOBJ-RESPQ
                                MQCO-NONE
                                W03-COMPCODE
                                W03-REASON.
      *
           IF W03-COMPCODE NOT = MQCC-OK
               MOVE 'MQCLOSE'            TO MO2-OPERATION
               MOVE W02-RESPONSE-QNAME   TO MO2-OBJECTNAME
               PERFORM RECORD-CALL-ERROR
               GO TO A-MAIN-EXIT
           END-IF.
      *
      *
       CLOSE-RESPONSE-Q-EXIT.
      *
      *    Return to performing section
      *
```

Figure 39 (Part 18 of 22). MVS WTPONG Cobol Program Source

```
          EXIT.
          EJECT
      *
      * ------------------------------------------------------------ *
       ASK-DATETIME SECTION.
      * ------------------------------------------------------------ *
      *
      *
          ACCEPT WO5-DATE FROM DATE.
          STRING  WO5-DD WO5-DATESEP WO5-MM WO5-DATESEP
                  '19' WO5-YY
                  DELIMITED BY SIZE INTO WO5-DATENEW.
          MOVE WO5-DATENEW TO WO5-DATE.
          ACCEPT WO5-TIME FROM TIME.
          STRING  WO5-HH WO5-TIMESEP WO5-MI WO5-TIMESEP
                  WO5-SS
                  DELIMITED BY SIZE INTO WO5-TIMENEW.
          MOVE WO5-TIMENEW TO WO5-TIME.
       ASK-DATETIME-EXIT.
          EXIT.
          EJECT
      *
      * ------------------------------------------------------------ *
       RECORD-CALL-ERROR SECTION.
      * ------------------------------------------------------------ *
      *                                                              *
      * This section writes an error message to sysout ddname        *
      * in server job                                                *
      *                                                              *
      * ------------------------------------------------------------ *
      *
      *
          PERFORM ASK-DATETIME.
          MOVE WO5-DATE TO MO2-DATE.
          MOVE WO5-TIME TO MO2-TIME.
      *
          MOVE WO3-COMPCODE    TO MO2-COMPCODE
          MOVE WO3-REASON      TO MO2-REASON
          MOVE MO2-DATE        TO MO3-DATE.
          MOVE MO2-TIME        TO MO3-TIME.
```

Figure 39 (Part 19 of 22). MVS WTPONG Cobol Program Source

```
      *
           MOVE MO2-CALL-ERROR-MSG TO OUTMSG.
           PERFORM DISPLAY-LOCAL.
       RECORD-CALL-ERROR-EXIT.
      *
      *    Return to performing section
      *
           EXIT.
           EJECT
      *
      * ------------------------------------------------------------ *
      *
      * ------------------------------------------------------------ *
       PUT-RESPONSE-MESSAGE SECTION.
      * ------------------------------------------------------------ *
      *                                                              *
      * Set the object descriptor, message descriptor and put message*
      *     options to the values required for creating a output     *
      *     message                                                  *
      *                                                              *
      * ------------------------------------------------------------ *
      *
           MOVE WO2-RESPONSE-QMGR  TO  MQOD-OBJECTQMGRNAME.
           MOVE WO2-RESPONSE-QNAME TO  MQOD-OBJECTNAME.
           MOVE MQMT-REPLY         TO  MQMD-MSGTYPE.
           MOVE MQRO-NONE          TO  MQMD-REPORT.
           MOVE SPACES             TO  MQMD-REPLYTOQ.
           MOVE SPACES             TO  MQMD-REPLYTOQMGR.
           MOVE WO3-REQ-MSGID      TO  MQMD-MSGID.
           MOVE WO3-REQ-CORRELID   TO  MQMD-CORRELID.
           MOVE WO5-DATETIME       TO  WO3-RSP-REQ-GET-DATETIME.
      *
           COMPUTE MQPMO-OPTIONS  =  MQPMO-NO-SYNCPOINT.
           MOVE WO3-HOBJ-REQUESTQ TO MQPMO-CONTEXT.
      *
           MOVE LENGTH OF WO3-PUT-BUFFER TO WO3-BUFFLEN.
      *
           CALL 'MQPUT' USING WO3-HCONN
                              WO3-HOBJ-RESPQ
                              MQMD
                              MQPMO
                              WO3-BUFFLEN
                              WO3-PUT-BUFFER
                              WO3-COMPCODE
                              WO3-REASON.
```

Figure 39 (Part 20 of 22). MVS WTPONG Cobol Program Source

```
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              MOVE 'MQPUT'          TO  MO2-OPERATION
              MOVE MQOD-OBJECTNAME  TO  MO2-OBJECTNAME
              PERFORM RECORD-CALL-ERROR
      *       PERFORM FORWARD-MSG-TO-DLQ
              GO   TO CLOSE-ALL
           END-IF.
      *
       PUT-RESPONSE-MESSAGE-EXIT.
           EXIT.
           EJECT
      *
       MOVE-REQ-TO-TS SECTION.
      * ------------------------------------------------------------ *
      *    Build TS Queue....from Request Q
      *        - Move    Message header       to  TS
      *        - Extract PUTDATE and PUTDATE from Request Queue to TS
      * ------------------------------------------------------------ *
      *
           MOVE  MQMD-MSGID       TO  WO3-TS-MSGID.
           MOVE  MQMD-CORRELID    TO  WO3-TS-CORRELID.
           MOVE  MQMD-PUTTIME     TO  WO3-TS-REQ-PUTTIME.
           MOVE  MQMD-PUTDATE     TO  WO3-TS-REQ-PUTDATE.
           MOVE  WO5-TIME         TO  WO3-TS-REQ-GETTIME.
           MOVE  WO5-DATE         TO  WO3-TS-REQ-GETDATE.
      *
           MOVE  WO3-REQ-MSG-TOT         TO  WO3-TS-MSG-TOT.
           MOVE  WO3-REQ-MSG-LEN         TO  WO3-TS-MSG-LEN.
           MOVE  WO3-REQ-MSG-TXT-50BYTE  TO  WO3-TS-MSG-TXT-50BYTE.
           MOVE  WO3-REQ-MSG-COMMENT     TO  WO3-TS-MSG-COMMENT.
      *
       MOVE-REQ-TO-TS-EXIT.
           EXIT.
           EJECT
      *
       MOVE-RSP-TO-TS SECTION.
      * ------------------------------------------------------------ *
      *    Build TS Queue....from ReSPONSE Q
      *        - Extract PUTDATE and PUTDATE
```

Figure 39 (Part 21 of 22). MVS WTPONG Cobol Program Source

```
      * ------------------------------------------------------------ *
      *
           MOVE  MQMD-PUTTIME   TO  WO3-TS-RSP-PUTTIME.
           MOVE  MQMD-PUTDATE   TO  WO3-TS-RSP-PUTDATE.
      *
       MOVE-RSP-TO-TS-EXIT.
           EXIT.
           EJECT
       DISPLAY-LOCAL  SECTION.
      * ------------------------------------------------------------ *
      *    DISPLAY Message to SYSOUT
      * ------------------------------------------------------------ *
      *
           DISPLAY    OUTMSG.
           MOVE SPACE TO OUTMSG.
      *
       DISPLAY-LOCAL-EXIT.
           EXIT.
       DISPLAY-CONS   SECTION.
      * ------------------------------------------------------------ *
      *    DISPLAY Message to Console
      * ------------------------------------------------------------ *
      *
           DISPLAY    OUTMSG UPON CONSOLE.
           MOVE SPACE TO OUTMSG.
      *
       DISPLAY-CONS-EXIT.
           EXIT.
           EJECT
      *
       FORWARD-MSG-TO-TSQ SECTION.
      * ------------------------------------------------------------ *
      *    Write extracted message to TS Queue
      * ------------------------------------------------------------ *
      *
           IF WO2-DEBUG NOT = SPACES THEN
             DISPLAY WO3-TS-REQ-RSP
           END-IF.
      *
       FORWARD-MSG-TO-TSQ-EXIT.
           EXIT.
           EJECT
      *
      * ------------------------------------------------------------ *
      *                    End of program                            *
      * ------------------------------------------------------------ *
      *
```

Figure 39 (Part 22 of 22). MVS WTPONG Cobol Program Source

C.4 WTPING/WTPONG Copy Books

```
      *
      *    PRGQO1X1
      *
      *    Request Queue message buffer format
           03  WO3-REQUEST.
               05  WO3-REQ-MSG-TOT         PIC 9(5).
               05  WO3-REQ-MSG-LEN         PIC 9(6).
               05  WO3-REQ-MSG-COMMENT     PIC X(65).
           03  WO3-REQ-MSG.
               05  WO3-REQ-MSG-SEG         PIC X(10)  OCCURS 1000.
      *
      *    PRGQO1X2
      *
      *    Response Queue message buffer format
           03  WO3-RSP-REQ-GET-DATETIME.
               05  WO3-RSP-REQ-GETDATE  PIC X(8).
               05  WO3-RSP-REQ-GETTIME  PIC X(8).
      *
      *
      *    PRGQO1X9
      *
      *
       01  MO2-CALL-ERROR-MSG.
           05  MO2-JOB              PIC X(04) VALUE 'SERV'.
           05                       PIC X(01) VALUE SPACE.
           05  MO2-TASK-NUMBER      PIC X(09) VALUE SPACE.
           05                       PIC X(01) VALUE SPACE.
           05  MO2-DATE             PIC X(08).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-TIME             PIC X(08).
           05                       PIC X(01) VALUE SPACE.
           05  MO2-OPERATION.
               10  MO2-OPER-MSG     PIC X(15).
               10                   PIC X(01) VALUE SPACE.
               10  MO2-OPER-MSGID   PIC X(8).
               10                   PIC X(01) VALUE SPACE.
               10  MO2-OPER-CORRID  PIC X(24).
           05                       PIC X(18) VALUE
               ' ERROR - COMPCODE:'.
           05  MO2-COMPCODE         PIC Z(08)9.
```

Figure 40 (Part 1 of 4). MVS WTPING/WTPONG Copy Books

```
           05                      PIC X(13) VALUE
               ' REASON CODE:'.
           05  MO2-REASON          PIC Z(08)9.
           05                      PIC X(12) VALUE
               ' OBJECTNAME:'.
           05  MO2-OBJECTNAME      PIC X(48).
      *
            01  MO3-CSML-ERROR-MSG.
           05                      PIC X(37) VALUE
             'An error has occurred in Batch SRVR  '.
           05  MO3-TRANSACTION     PIC X(04) VALUE 'SERV'.
           05                      PIC X(10) VALUE
             '          '.
           05  MO3-TASK-NUMBER     PIC X(07) VALUE SPACE.
           05                      PIC X(01) VALUE SPACE.
           05  MO3-DATE            PIC X(08).
           05                      PIC X(01) VALUE SPACE.
           05  MO3-TIME            PIC X(08).
           05                      PIC X(40) VALUE
             ' View Sysout or console log   ''       '''.
      *
            01  MO4-STARTUP-ERROR       PIC X(60) VALUE
             ' started without data, a MQTM structure was expected.'.
      *
      *     PRGTRAN
      *
      *
      *   MISCELLANEOUS FIELDS
      *
            01  WS-SUB                      PIC 9(9) COMP.
            01  TRAN-BUF-LEN                PIC 9(9) COMP.
            01  TRAN-BUFFER.
           05 TRAN-ARRAY OCCURS 100000 PIC X.
            01  LS-TRAN-BUF-PTR             USAGE POINTER.
            01  WS-ASCII-PTR                USAGE POINTER.
            01  WS-EBCDIC-PTR               USAGE POINTER.
      *
            01  ASCII-CODE-PAGE.
           05 FILLER PIC X(16) VALUE
               X'3A3B0203040506070809A0B0C0D0E0F'.
```

Figure 40 (Part 2 of 4). MVS WTPING/WTPONG Copy Books

```
           05 FILLER PIC X(16) VALUE
               X'101112131415161718191A1B1C1D1E1F'.
           05 FILLER PIC X(16) VALUE
               X'202122232425262728292A2B2C2D2E2F'.
           05 FILLER PIC X(16) VALUE
               X'303132333435363738393A3B3C3D3E3F'.
           05 FILLER PIC X(16) VALUE
               X'204142434445464748494A4B4C4D4E4F'.
           05 FILLER PIC X(16) VALUE
               X'505152535455565758595A5B2A5D5E5F'.
           05 FILLER PIC X(16) VALUE
               X'2D616263646566768696A6B6C6D6E6F'.
           05 FILLER PIC X(16) VALUE
               X'70717273747576777879 7A7B7C277E7F'.
           05 FILLER PIC X(16) VALUE
               X'80616263646566768698A8B8C8D8E8F'.
           05 FILLER PIC X(16) VALUE
               X'906A6B6C6D6E6F7071729A9B9C9D9E9F'.
           05 FILLER PIC X(16) VALUE
               X'A0A1737475767778797AAAABACADAEAF'.
           05 FILLER PIC X(16) VALUE
               X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
           05 FILLER PIC X(16) VALUE
               X'C041424344454647 4849CACBCCCDCECF'.
           05 FILLER PIC X(16) VALUE
               X'D04A4B4C4D4E4F505152DADBDCDDDEDF'.
           05 FILLER PIC X(16) VALUE
               X'E0E1535455565758595AEAEBECEDEEEF'.
           05 FILLER PIC X(16) VALUE
               X'303132333435363738 39FAFBFCFDFEFF'.
      *
      *   ASCII CODE TABLE
      *
       01 LS-ASCII-TABLE REDEFINES ASCII-CODE-PAGE.
          03  LS-ASCII-ENTRY         PIC X OCCURS 256
                INDEXED BY WS-ASCII-INDX.
      *
       01  EBCDIC-CODE-PAGE.
           05 FILLER PIC X(16) VALUE
               X'7A5E0203040506070809 0A0B0C0D0E0F'.
```

Figure 40 (Part 3 of 4). MVS WTPING/WTPONG Copy Books

```
           05 FILLER PIC X(16) VALUE
               X'101112131415161718191A1B1C1D1E1F'.
           05 FILLER PIC X(16) VALUE
               X'40212223246C26274D5D2A2B2C2D4B61'.
           05 FILLER PIC X(16) VALUE
               X'F0F1F2F3F4F5F6F7F8F93A3B3C3D3E3F'.
           05 FILLER PIC X(16) VALUE
               X'40C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'.
           05 FILLER PIC X(16) VALUE
               X'D7D8D9E2E3E4E5E6E7E8E95B5C5D5E6D'.
           05 FILLER PIC X(16) VALUE
               X'60818283848586878889919293949596'.
           05 FILLER PIC X(16) VALUE
               X'979899A2A3A4A5A6A7A8A94D5D7D7E7F'.
           05 FILLER PIC X(16) VALUE
               X'80818283848586878889 8A8B8C8D8E8F'.
           05 FILLER PIC X(16) VALUE
               X'909192939495969798999A9B9C9D9E9F'.
           05 FILLER PIC X(16) VALUE
               X'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'.
           05 FILLER PIC X(16) VALUE
               X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
           05 FILLER PIC X(16) VALUE
               X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'.
           05 FILLER PIC X(16) VALUE
               X'D0D1D0D3D4D5D6D7D8D9DADBDCDDDEDF'.
           05 FILLER PIC X(16) VALUE
               X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'.
           05 FILLER PIC X(16) VALUE
               X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'.
      *
      *   EBCDIC CODE TABLE
      *
       01 LS-EBCDIC-TABLE REDEFINES EBCDIC-CODE-PAGE.
          03  LS-EBCDIC-ENTRY        PIC X OCCURS 256
                INDEXED BY WS-EBCDIC-INDX.
```

Figure 40 (Part 4 of 4). MVS WTPING/WTPONG Copy Books

C.5 Compile JCL for MVS Batch Applications

```
//COMPPRG JOB 'MQ SAMPLE',MSGCLASS=6,CLASS=I,NOTIFY=WTWKSH1
//*-------------------------------------------------------------------*
//*
//*   COMPILE MQSERIES BATCH SERVER PROGRAMS
//*
//*
//COMPMQS  PROC OUTC='*',
//       PGMMEM=' ',
//       REG=2048K,
//       LNKPARM='LIST,XREF',
//       WORK=SYSDA
//COB    EXEC PGM=IGYCRCTL,REGION=&REG,
//       PARM='LIB,OBJECT,RENT,RES,APOST,MAP,XREF'
//STEPLIB  DD DSN=COB2.COB2COMP,DISP=SHR
//SYSLIB   DD DSN=MQM.V1R3M0.SCSQCOBC,DISP=SHR  /* MQM MVS/ESA LIB */
//         DD DSN=MQM.USER.SOURCE,DISP=SHR        /* COBOL COPYBOOKS */
//SYSPRINT DD SYSOUT=&OUTC
//SYSIN    DD DSN=MQM.USER.SOURCE(&PGMMEM),DISP=SHR
//SYSLIN   DD DSN=&&LOADSET,DISP=(MOD,PASS),
//            UNIT=&WORK,SPACE=(80,(250,100))
//SYSUT1   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT2   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT3   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT4   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT5   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT6   DD UNIT=&WORK,SPACE=(460,(350,100))
//SYSUT7   DD UNIT=&WORK,SPACE=(460,(350,100))
//*
//LKED   EXEC PGM=IEWL,REGION=&REG,
//            PARM='&LNKPARM',COND=(5,LT,COB)
//SYSLIB   DD DSN=COB2.COB2LIB,DISP=SHR
//         DD DSN=MQM.USER.LOAD,DISP=SHR
//CSQSTUB  DD DSN=MQM.V1R3M0.SCSQLOAD,DISP=SHR       /* MQM MVS/ESA */
//SYSLMOD  DD DSN=MQM.USER.LOAD(&PGMMEM),DISP=SHR
//SYSUT1   DD UNIT=&WORK,DCB=BLKSIZE=1024,
//            SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=&OUTC
//SYSLIN   DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD DDNAME=SYSIN
// PEND
//STEP1    EXEC COMPMQS,
//         PGMMEM=PRGQJIW1
//LKED.SYSIN DD *
   INCLUDE CSQSTUB(CSQBSTUB)
   NAME    PRGQJIW1(R)
/*
//* PRGQJPS8
```

Figure 41. Compile JCL

C.6 Run Job Web WTPING

```
//WTWKSH1J JOB 'MQ SAMPLE',MSGCLASS=6,CLASS=I,NOTIFY=WTWKSH1
//*-------------------------------------------------------------------*
//*
//*   RUN WEB CLIENT ON SA25/18
//*
//*   THIS JOB HANDLES THE WTPING CLIENT AND GETS THE
//*   INPUT PARAMETERS FROM THE WEBBROWSER VIA
//*   MQSERIES INTERNET GATEWAY
//*
//RUNCLNT EXEC PGM=PRGQJIW1,REGION=4M,
//             PARM='CSQ2,WEB.CLIENT.QUEUE'
//STEPLIB  DD DSN=MQM.USER.LOAD,DISP=SHR
//         DD DSN=COB2.COB2LIB,DISP=SHR
//         DD DSN=MQM.V1R4M0.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

Figure 42. Run Job Web WTPING

C.7 Run Job Web WTPONG Server

```
//WTWKSH2J JOB 'MQ SAMPLE',MSGCLASS=6,CLASS=I,NOTIFY=WTWKSH2
//*-------------------------------------------------------------------*
//*
//*   RUN BATCH SERVER ON SA18
//*
//*   BACKEND MESSAGE SERVER TO  WTPING LIKE CLIENT
//*
//*
//RUNSERV EXEC PGM=PRGQJPS8,REGION=4M,
//             PARM='CSQ2,DXB.CSQ2.REQUESTQ'
//* X = DEBUG MODE
//*            PARM='CSQ2,DXB.CSQ2.REQUESTQ,X'
//STEPLIB  DD DSN=MQM.USER.LOAD,DISP=SHR
//         DD DSN=COB2.COB2LIB,DISP=SHR
//         DD DSN=MQM.V1R4M0.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
```

Figure 43. Run Job Web WTPONG Server

Appendix D. Source Listing and Compile Procedures

This appendix contains source listings and procedures for MVS and OS/2.

D.1 HTML for the Web File Browser Example

```
<HEAD>
<TITLE>MQSeries Browse Remote MGMT-Files </TITLE>
</HEAD>
<BODY BGCOLOR="#E0E0FF">
<center>
<A HREF="./MQGate.html">
<img src="./images/MQGate.gif" height=124 width=435
alt="[ MQGate Page ]" border=0>
</A>
<hr noshade size=1 width=545 align=center>
</center>
<FORM ACTION="/cgi-bin/MQGate" METHOD="POST">
<P>This is a sample to start a MVS application to read a file and send it back
HTML.You will need a FORM capable browser.</P>
<INPUT NAME="MQIGwQueueManager" TYPE="hidden" VALUE="">
<INPUT NAME="MQIGwQueue" TYPE="hidden" VALUE="WEB.MGMT.QUEUE">
<p>
Choose the right filename to read:
<pre>
    <input type="radio" name="FileName" value="F1" CHECKED > DSNAME ABC
    <input type="radio" name="FileName" value="XX" > Terminate Server Program
<P>Click on "SUBMIT" to
read the file and send records as messages to the browser.</P>
<hr noshade size=1 width=545 align=center>
<P>
<font size=+1>
<INPUT TYPE="submit" VALUE="SUBMIT">
</font>
</P>
</FORM>
<hr noshade size=1 width=545 align=center>
</BODY>
</HTML>
```

Figure 44. HTML WTMQCAT

D.2 Cobol Source of the MVS File Browser Program

```
CBL NODYNAM,LIB,OBJECT,RENT,RES,APOST
      * ------------------------------------------------------------ *
       IDENTIFICATION DIVISION.
      * ------------------------------------------------------------ *
       PROGRAM-ID. PRGQJIW2.
      *REMARKS
      * ------------------------------------------------------------ *
      * ------------------------------------------------------------ *
      * BATCH DRIVEN MQSERIES Server PROGRAM------------------------ *
      *
      * THIS PROGRAM TAKES DATA FROM A PARM CARD AS WELL
      * FROM A MSGQ WEB.MGMT.QUEUE
      * IT IS A FILE SERVER PROGRAM TO A WEB BROWSER APPLICATIONS.
      * IT SENDS THE CONTENT OF A FILE BACK IN ONE MESSAGE
      * TO THE WEB BROWSER APPLICATION.
      *
      *
      *
      * ------------------------------------------------------------ *
      * ------------------------------------------------------------ *
      * CLIENT PROGRAM FOR SUBAREA SA25/18
      * NAME OF THE PROGRAM: PRGQJ I W   2:
      *                      !     ! !   !
      *                      !     ! !   +---> TEAM NAME (NOW FIXED)
      *                      !     ! +-------> CLIENT PROGRAM
      *                      !     !
      *                      !     +---------> S.A. 25
      *                      +---------------> FIXED IDENTIFIER
      *
      * PARM   (REQ  )   QUEUE: WEB.MGMT.QUEUE(FIXED)
      *                              +--------> TARGET Q MANAGER NAME
      *                                         FROM USER INPUT
      *
      * ------------------------------------------------------------ *
      * ----HOW THE INPUT PARMS ARE SPECIFIED ---------------------- *
      * ------------------------------------------------------------ *
      *                    ---------------
      *START.
      *    GET WEB.MGMT.QUEUE GET FILENAME TO PROCESS
      *
```

Figure 45 (Part 1 of 14). Cobol Source of the MVS File Browser Program

```
*                   PERFORM OPEN FILE
*                   PERFORM READ-FILE AND CREATE MESSAGE CONTENT
*                   PERFORM SEND MESSAGE BACK TO CLIENT
* ------------------------------------------------------------- *
 ENVIRONMENT DIVISION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT MGMT1  ASSIGN TO MGMTIN01.
* ------------------------------------------------------------- *
 DATA DIVISION.
 FILE SECTION.
 FD MGMT1
     RECORD CONTAINS 80 CHARACTERS
     BLOCK CONTAINS 0
     RECORDING MODE F
     LABEL RECORDS ARE STANDARD
     DATA RECORD IS PARM-MGMT01.
 01  PARM-MGMT01.
     05 FILLER   PIC X(80).
* ------------------------------------------------------------- *
* ------------------------------------------------------------- *
 WORKING-STORAGE SECTION.
* ------------------------------------------------------------- *
*
*    W00 - GENERAL WORK FIELDS
*
 01  OUTMSG                      PIC X(80) VALUE SPACE.
 01  W00-PARM-BUFFER.
     05  W00-PARM-HDR     PIC X(54).
     05  W00-FILE-STRING  PIC X(2).
     05  FILLER           PIC X(100).
*
*    W02 - QUEUE NAME FIELD
*
 01  W02-MQM                     PIC X(48) VALUE SPACE.
 01  W02-CLIENT-QNAME            PIC X(48) VALUE SPACE.
 01  EOF-FLAG                    PIC X(01) VALUE SPACE.
 01  TERM-MSG                    PIC X(01) VALUE SPACE.
*
*    QUEUE MANAGER NAMES
```

Figure 45 (Part 2 of 14). Cobol Source of the MVS File Browser Program

```
      * ADD HERE NEW QUEUE MANAGER NAMES
      *
       01  WO2-QMGR-NAME.
           03  QMGR-CSQ1               PIC X(48)
           VALUE 'CSQ1                                          '.
           03  QMGR-CSQ2               PIC X(48)
           VALUE 'CSQ2                                          '.
      *
      *    WO3 - MQM API FIELD
      *
       01  WO3-API-FILED.
           03  WO3-COMPCODE            PIC S9(9)  BINARY VALUE 0.
           03  WO3-REASON              PIC S9(9)  BINARY VALUE 0.
           03  WO3-OPTIONS             PIC S9(9)  BINARY.
           03  WO3-HCONN               PIC S9(9)  BINARY.
           03  WO3-HOBJ-CLIENTQ        PIC S9(9)  BINARY.
           03  WO3-HOBJ-CLIENT-RESPQ   PIC S9(9)  BINARY.
           03  WO3-CLIENT-CORRELID     PIC X(24)  VALUE  SPACES.
           03  WO3-CLIENT-MSGID        PIC X(24)  VALUE  SPACES.
           03  WO3-CLIENT-RESP-QNAME   PIC X(48)  VALUE  SPACES.
           03  WO3-CLIENT-RESP-QMGR    PIC X(48)  VALUE  SPACES.
           03  WO3-DATALEN             PIC S9(9)  BINARY.
           03  WO3-BUFFLEN             PIC S9(9)  BINARY.
           03  WO3-SELECTORCOUNT       PIC S9(9)  BINARY VALUE 1.
           03  WO3-INTATTRCOUNT        PIC S9(9)  BINARY VALUE 1.
           03  WO3-CHARATTRLENGTH      PIC S9(9)  BINARY VALUE 0.
           03  WO3-MAXMSGS             PIC S9(9)  BINARY VALUE 1000.
           03  WO3-MSGS                PIC S9(9)  BINARY VALUE 0.
           03  WO3-CHARATTRS           PIC X(48)  VALUE  LOW-VALUES.
           03  WO3-NEWLINE             PIC X(1)  VALUE  X'0A'.
           03  WO3-END-MSG             PIC X(80) VALUE
               'SERVER TERMINATED SUCCESSFULLY'.
           03  WO3-SELECTOR-TABLE.
               05  WO3-SELECTORS       PIC S9(9)  BINARY OCCURS 2.
           03  WO3-INTATTRS-TABLE.
               05  WO3-INTATTRS        PIC S9(9)  BINARY OCCURS 2.
```

Figure 45 (Part 3 of 14). Cobol Source of the MVS File Browser Program

```
*
* Some of the following data would not show correctly when
* this redbook is viewed online or via CD.  The field has
* ASCII representation, their hex values being:
*
*  ASCII
*  text   C o n t e n t - T y p e :   t e x t / h t m l
*  (26 characters from the source code program)
*
*   Hex   4 6 6 7 6 6 7 2 5 7 7 6 3 2 7 6 7 7 2 6 7 6 6 0 0 0
*  Value  3 F E 4 5 E 4 D 4 9 0 5 A 0 4 5 8 4 F 8 4 D C A A A
*
*  - OR -
*
*  ASCII
*  text   C o n t e n t - T y p e :     t e x t / p l a i n
*  (28 characters of 33 defined in source code program)
*   Hex   4 6 6 7 6 6 7 2 5 7 7 6 3 2 2 7 6 7 7 2 7 6 6 6 6 0 0 0
*  Value  3 F E 4 5 E 4 D 4 9 0 5 A 0 0 4 5 8 4 F 0 C 1 9 E A A A
*
*  (characters of 29 - 33 from the source code program)
*   Hex   7 6 7 7 2
*  Value  4 5 8 4 0
*
*  ASCII  t e x t
*  text
*
 01  WO3-REPLY-CLIENT-MSG.
     03  WO3-REPLY-HTML          PIC X(34) VALUE
* Content-Type: text/plain or html
         '***  See Above Comment   ********'.
     03  WO3-REPLY-CLIENT-MSG-TABLE.
        05 WO3-REPLY-ARRAY OCCURS 1000 INDEXED BY WO3-I1.
```

Figure 45 (Part 4 of 14). Cobol Source of the MVS File Browser Program

```
           07  FILLER                  PIC X(81) VALUE SPACE.
      ****************************************************************
      *
      *
      *    API CONTROL BLOCK
      *
       01  MQM-OBJECT-DESCRIPTOR.
           COPY CMQODV.
       01  MQM-MESSAGE-DESCRIPTOR.
           COPY CMQMDV.
       01  MQM-GET-MESSAGE-OPTIONS.
           COPY CMQGMOV.
       01  MQM-PUT-MESSAGE-OPTIONS.
           COPY CMQPMOV.
      *
      *    MQV CONTAINS CONSTANTS AND RETURN CODES
      *
       01  MQM-CONST-RC.
           COPY CMQV   SUPPRESS.
      *
      * ---------------------------------------------------------- *
       EJECT
       COPY PRGTRAN.
       EJECT
       LINKAGE SECTION.
       01  PARMDATA.
           05  PARM-LEN               PIC S9(03) BINARY.
           05  PARM-STRING            PIC X(100).
      *
      * ---------------------------------------------------------- *
       PROCEDURE DIVISION USING PARMDATA.
      * ---------------------------------------------------------- *
      * ---------------------------------------------------------- *
       A-MAIN SECTION.
      * ---------------------------------------------------------- *
      *
      * THIS SECTION INTERPRETS THE PARM CARDS
      * AND SELECTS QUEUE-MANAGER
      *
      * ---------------------------------------------------------- *
```

Figure 45 (Part 5 of 14). Cobol Source of the MVS File Browser Program

```
 PARM-INFO-SELECT.
*
*    IF NO DATA WAS PASSED, CREATE A MESSAGE, PRINT IT, AND
*    EXIT
*
     IF PARM-LEN = 0 THEN
        STRING 'MQS95001 : 95001  '
               'NO PARM-CARD PASSED '
                DELIMITED BY SIZE INTO OUTMSG
        PERFORM DISPLAY-CONS
        GO TO A-MAIN-EXIT
     END-IF.
*
*    SEPARATE INTO THE RELEVANT FIELDS ANY DATA PASSED IN THE
*    PARM STATEMENT
*
     UNSTRING PARM-STRING DELIMITED BY ALL ','
                             INTO WO2-MQM
                                  WO2-CLIENT-QNAME.
*
*    CHECK FOR VALID QUEUEMANAGER AND QUEUE COMBINATION
*
     IF WO2-MQM = 'CSQ1' THEN GO TO APPL-CONNECT-QMGR.
     IF WO2-MQM = 'CSQ2' THEN GO TO APPL-CONNECT-QMGR.
     STRING 'MQS95001 : 95001           '
            'NO VALID QUEUEMANAGER IN PARM '
            DELIMITED BY SIZE INTO OUTMSG
       PERFORM DISPLAY-CONS
       GO   TO A-MAIN-EXIT.
 APPL-CONNECT-QMGR.

* ------------------------------------------------------------ *
*                                                              *
* THIS SECTION CONNECTS THE SERVER TO QMGR                     *
*                                                              *
* ------------------------------------------------------------ *
     IF WO2-CLIENT-QNAME = SPACE THEN
         STRING 'MQS95001 : 95001           '
                'NO VALID CLIENT QNAME IN PARM '
                DELIMITED BY SIZE INTO OUTMSG
```

Figure 45 (Part 6 of 14). Cobol Source of the MVS File Browser Program

```
           PERFORM DISPLAY-CONS
           GO   TO A-MAIN-EXIT.
*
*
*    CONNECT TO THE SPECIFIED QUEUE MANAGER.
*
     CALL 'MQCONN' USING WO2-MQM
                         WO3-HCONN
                         WO3-COMPCODE
                         WO3-REASON.
*
*    TEST THE OUTPUT OF THE CONNECT CALL.  IF THE CALL FAILED,
*    PRINT AN ERROR MESSAGE SHOWING THE COMPLETION CODE AND
*    REASON CODE
*
     IF WO3-COMPCODE NOT = MQCC-OK
        MOVE 'MQS95001 : MQS95001 CONNECT ERROR' TO OUTMSG
        PERFORM DISPLAY-CONS
        GO   TO A-MAIN-EXIT
     END-IF.
     IF WO2-MQM = 'CSQ1' THEN
       STRING 'MQS95001 : 95001 SA25 CSQ1 '
                'CLIENT CONNECTED '
              DELIMITED BY SIZE INTO OUTMSG
      ELSE
       STRING 'MQS95001 : 95001 SA18 CSQ2 '
                'CLIENT CONNECTED '
              DELIMITED BY SIZE INTO OUTMSG
       PERFORM DISPLAY-CONS
      END-IF.
 MAIN-PROCESS SECTION.
*
*
*   GET  PARMIN MESSAGE
*
*
 PROCESS-CLIENT-MSG.
     PERFORM OPEN-CLIENT-PARMS.
     MOVE SPACE TO TERM-MSG.
     PERFORM WITH TEST BEFORE UNTIL (TERM-MSG NOT = SPACE)
```

Figure 45 (Part 7 of 14). Cobol Source of the MVS File Browser Program

```
            PERFORM GET-CLIENT-PARMS
            PERFORM PUT-MSG-TO-CLIENT
        END-PERFORM.
        PERFORM CLOSE-CLIENT-PARMS.
 A-MAIN-EXIT.
        GOBACK.
        EJECT
 DISPLAY-LOCAL  SECTION.
* ------------------------------------------------------------- *
*    DISPLAY MESSAGE ON SYSOUT ONLY ON DEBUG MODE
* ------------------------------------------------------------- *
*
          DISPLAY    OUTMSG.
          MOVE SPACE TO OUTMSG.
*
 DISPLAY-LOCAL-EXIT.
        EXIT.
 DISPLAY-CONS   SECTION.
* ------------------------------------------------------------- *
*    DISPLAY MESSAGE TO CONSOLE
* ------------------------------------------------------------- *
*
        DISPLAY    OUTMSG UPON CONSOLE.
        MOVE SPACE TO OUTMSG.
*
 DISPLAY-CONS-EXIT.
        EXIT.
        EJECT
 OPEN-CLIENT-PARMS SECTION.
* ------------------------------------------------------------- *
*    GET PARM MESSAGE FROM WEB CLIENT
* ------------------------------------------------------------- *
*
        MOVE MQOT-Q            TO MQOD-OBJECTTYPE.
        MOVE WO2-CLIENT-QNAME  TO MQOD-OBJECTNAME.
        MOVE ZERO TO WO3-HOBJ-CLIENTQ.
*
        MOVE ZERO TO WO3-OPTIONS.
        COMPUTE WO3-OPTIONS    =  MQOO-INPUT-SHARED.
*
```

Figure 45 (Part 8 of 14). Cobol Source of the MVS File Browser Program

```
          CALL 'MQOPEN' USING W03-HCONN
                              MQOD
                              W03-OPTIONS
                              W03-HOBJ-CLIENTQ
                              W03-COMPCODE
                              W03-REASON.
      *
          IF W03-COMPCODE NOT = MQCC-OK
             DISPLAY 'MQOPEN CLIENTQ ERROR'
             DISPLAY W03-COMPCODE, W03-REASON
             GO TO A-MAIN-EXIT
          END-IF.
       OPEN-CLIENT-PARMS-EXIT.
          EXIT.
       GET-CLIENT-PARMS SECTION.
          COMPUTE MQGMO-OPTIONS = MQGMO-WAIT +
                                  MQGMO-NO-SYNCPOINT.
      *
      *
      *  WAIT UNLIMITED FOR REQUESTS FROM WEBCLIENT
      *
      *
          MOVE MQWI-UNLIMITED     TO MQGMO-WAITINTERVAL.
      *
          MOVE SPACE TO W00-PARM-BUFFER.
          MOVE LENGTH OF W00-PARM-BUFFER TO W03-BUFFLEN.
          MOVE MQMI-NONE                TO MQMD-MSGID.
          MOVE MQCI-NONE                TO MQMD-CORRELID.
      *
      *    MAKE THE MQGET CALL TO CLIENT INPUT QUEUE
      *
          CALL 'MQGET' USING W03-HCONN
                             W03-HOBJ-CLIENTQ
                             MQMD
                             MQGMO
                             W03-BUFFLEN
                             W00-PARM-BUFFER
                             W03-DATALEN
                             W03-COMPCODE
                             W03-REASON.
```

Figure 45 (Part 9 of 14). Cobol Source of the MVS File Browser Program

```
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              DISPLAY 'MQGET  CLIENTQ ERROR'
              DISPLAY WO3-COMPCODE, WO3-REASON
              GO TO A-MAIN-EXIT
           END-IF.
           IF WO3-DATALEN < 56 THEN
              DISPLAY 'WRONG CLIENT MESSAGE'
              MOVE 'X' TO TERM-MSG
              GO TO A-MAIN-EXIT
           END-IF.

      *
            MOVE MQMD-MSGID     TO  WO3-CLIENT-MSGID
            MOVE MQMD-CORRELID  TO  WO3-CLIENT-CORRELID
            MOVE MQMD-REPLYTOQ  TO  WO3-CLIENT-RESP-QNAME
            MOVE MQMD-REPLYTOQMGR TO WO3-CLIENT-RESP-QMGR
            DISPLAY WOO-PARM-BUFFER
            PERFORM BUILD-MESSAGE.
      *
      *
       GET-CLIENT-PARMS-EXIT.
           EXIT.
       CLOSE-CLIENT-PARMS SECTION.
      *
      *
      *
           CALL 'MQCLOSE' USING WO3-HCONN
                                WO3-HOBJ-CLIENTQ
                                MQCO-NONE
                                WO3-COMPCODE
                                WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              DISPLAY 'MQCLOSE CLIENTQ ERROR'
              DISPLAY WO3-COMPCODE, WO3-REASON
              GO   TO A-MAIN-EXIT
           END-IF.
      *
      *
```

Figure 45 (Part 10 of 14). Cobol Source of the MVS File Browser Program

```
 CLOSE-CLIENT-PARMS-EXIT.
     EXIT.
     EJECT
 PUT-MSG-TO-CLIENT SECTION.
* ------------------------------------------------------------ *
*    SEND MESSAGE BACK TO CLIENT
* ------------------------------------------------------------ *
     MOVE MQOT-Q                 TO  MQOD-OBJECTTYPE.
     MOVE WO3-CLIENT-RESP-QNAME  TO  MQOD-OBJECTNAME.
     MOVE WO3-CLIENT-RESP-QMGR   TO  MQOD-OBJECTQMGRNAME.

*
     COMPUTE WO3-OPTIONS = MQOO-OUTPUT
*
     MOVE ZERO  TO  WO3-HOBJ-CLIENT-RESPQ
*
     CALL 'MQOPEN'  USING WO3-HCONN
                          MQOD
                          WO3-OPTIONS
                          WO3-HOBJ-CLIENT-RESPQ
                          WO3-COMPCODE
                          WO3-REASON.
*
     IF WO3-COMPCODE NOT = MQCC-OK
        DISPLAY 'MQOPEN  REPLYQ  ERROR'
        DISPLAY WO3-COMPCODE, WO3-REASON
        GO   TO A-MAIN-EXIT
     END-IF.
        COMPUTE MQPMO-OPTIONS = MQPMO-NO-SYNCPOINT.
        MOVE WO3-CLIENT-MSGID    TO MQMD-CORRELID.
        MOVE MQMI-NONE           TO MQMD-MSGID.
*
        MOVE ZERO TO WO3-DATALEN.
*
        COMPUTE WO3-DATALEN = 34 + (WO3-MSGS * 81).
*
*
* SPECIFY THE FORMAT AS STRING
* MQGate expects this!
*
```

Figure 45 (Part 11 of 14). Cobol Source of the MVS File Browser Program

```
      *
             MOVE 273          TO MQMD-ENCODING.
             MOVE 850          TO MQMD-CODEDCHARSETID.
      *
      *     EBCDIC TO ASCII
      *
             MOVE SPACE TO TRAN-BUFFER.
             MOVE LENGTH OF WO3-REPLY-CLIENT-MSG-TABLE
             TO TRAN-BUF-LEN.
             MOVE WO3-REPLY-CLIENT-MSG-TABLE TO TRAN-BUFFER.
             PERFORM TRAN-MSG.
             MOVE TRAN-BUFFER TO WO3-REPLY-CLIENT-MSG-TABLE.
      *
      *
             CALL 'MQPUT'  USING  WO3-HCONN
                                  WO3-HOBJ-CLIENT-RESPQ
                                  MQMD
                                  MQPMO
                                  WO3-DATALEN
                                  WO3-REPLY-CLIENT-MSG
                                  WO3-COMPCODE
                                  WO3-REASON.
           IF WO3-COMPCODE NOT = MQCC-OK
              DISPLAY 'MQPUT   REPLYQ  ERROR'
              DISPLAY WO3-COMPCODE, WO3-REASON
              GO   TO A-MAIN-EXIT
           END-IF.
           MOVE SPACE TO WO3-REPLY-CLIENT-MSG.
      *
      *    CLOSE REPLY QUEUE TO CLIENT
      *
           CALL 'MQCLOSE' USING WO3-HCONN
                                WO3-HOBJ-CLIENT-RESPQ
                                MQCO-NONE
                                WO3-COMPCODE
                                WO3-REASON.
      *
           IF WO3-COMPCODE NOT = MQCC-OK
              DISPLAY 'MQCLOSE REPLYQ  ERROR'
              DISPLAY WO3-COMPCODE, WO3-REASON
```

Figure 45 (Part 12 of 14). Cobol Source of the MVS File Browser Program

```
              GO   TO A-MAIN-EXIT
          END-IF.
      *
      *
      *
       PUT-MSG-TO-CLIENT-EXIT.
          EXIT.
      *
       BUILD-MESSAGE SECTION.
      * ------------------------------------------------------------ *
      *
          EVALUATE TRUE
              WHEN WOO-FILE-STRING = 'F1'
                   PERFORM READ-MGMTO1
      *     LOOK FOR TERMINATION STRING FROM CLIENT
              WHEN WOO-FILE-STRING = 'XX'
                   MOVE 'X' TO TERM-MSG
                   MOVE 1 TO WO3-MSGS
                   SET WO3-I1 TO 1
                   MOVE SPACE TO WO3-REPLY-CLIENT-MSG-TABLE
                   MOVE 'SERVER TERMINATED' TO WO3-END-MSG
                   STRING WO3-END-MSG WO3-NEWLINE
                       DELIMITED BY SIZE INTO
                       WO3-REPLY-ARRAY(WO3-I1)
                     END-STRING
              WHEN OTHER
                   DISPLAY 'INVALID PARAMETER FROM CLIENT'
                   GO TO A-MAIN-EXIT
          END-EVALUATE.
      *

       BUILD-MESSAGE-EXIT.
          EXIT.
      *
       READ-MGMTO1   SECTION.
      * ------------------------------------------------------------ *
      *
      *
          MOVE SPACE TO EOF-FLAG.
          OPEN INPUT MGMT1.
```

Figure 45 (Part 13 of 14). Cobol Source of the MVS File Browser Program

```
           SET WO3-I1 TO 1.
           MOVE ZERO TO WO3-MSGS.
           MOVE SPACE TO WO3-REPLY-CLIENT-MSG-TABLE.
      *
           PERFORM WITH TEST BEFORE UNTIL (EOF-FLAG NOT = SPACE)
                READ MGMT1 AT END
                            MOVE 'X' TO EOF-FLAG
                END-READ
                IF EOF-FLAG = SPACE THEN
                   IF WO3-I1 > WO3-MAXMSGS THEN
                      MOVE 'X' TO EOF-FLAG
                   ELSE
                      ADD 1 TO WO3-MSGS
                      STRING PARM-MGMTO1 WO3-NEWLINE
                        DELIMITED BY SIZE INTO
                        WO3-REPLY-ARRAY(WO3-I1)
                      END-STRING
                      SET WO3-I1 UP BY 1
                   END-IF
                END-IF
           END-PERFORM.
      *
           CLOSE MGMT1.
       READ-MGMTO1-EXIT.
           EXIT.
       TRAN-MSG  SECTION.
      * ------------------------------------------------------------ *
      *
      *   PERFORM THE RELEVANT CONVERSION
      *   NOTE THAT IF WE DON'T FIND A MATCH ON ANY CHARACTER,
      *   WE LEAVE THE DATA ASIS
      *
      *   CHANGE 437 TO 850 CCSID
      *
           PERFORM VARYING WS-SUB FROM 1   BY 1
                   UNTIL WS-SUB > TRAN-BUF-LEN
                      SET WS-EBCDIC-INDX TO 1
                      SEARCH LS-EBCDIC-ENTRY
                        WHEN LS-EBCDIC-ENTRY(WS-EBCDIC-INDX)
                          = TRAN-ARRAY(WS-SUB)
                            MOVE LS-ASCII-ENTRY(WS-EBCDIC-INDX)
                             TO TRAN-ARRAY(WS-SUB)
                      END-SEARCH
           END-PERFORM.
       TRAN-MSG-EXIT.
           EXIT.
```

Figure 45 (Part 14 of 14). Cobol Source of the MVS File Browser Program

D.3 JCL to Run MVS File Browser Program

```
//WTWKSH1J JOB 'MQ SAMPLE',MSGCLASS=6,CLASS=I,NOTIFY=WTWKSH1
//*-------------------------------------------------------------------*
//*
//*   RUN WEB FILE SERVER ON SA25/18
//*
//*   THIS SERVER PROGRAM RECEIVES A MESSAGE FROM
//*   THE WEBBROWSER VIA MQSERIES INTERNET GATEWAY
//*   AND SENDS BACK THE FILE SPECIFIED UNDER
//*   DDNAME MGMTINO1
//*
//*   IT COULD BE SAMPLE TO GET A MANAGEMENT LOGFILE
//*   DOWN TO A WEBBROWSER!
//*
//RUNCLNT EXEC PGM=PRGQJIW2,REGION=4M,
//             PARM='CSQ2,WEB.MGMT.QUEUE'
//STEPLIB  DD DSN=MQM.USER.LOAD,DISP=SHR
//         DD DSN=COB2.COB2LIB,DISP=SHR
//         DD DSN=MQM.V1R4MO.SCSQAUTH,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//*
//*  ======>   INPUT DATASET SHOWN TO THE WEBBROWSER
//*
//MGMTINO1 DD DISP=SHR,DSN=MQM.USER.SOURCE(TEST)
```

Figure 46. JCL to Run MVS File Browser Program

D.4 Translation Table Copy Structure

```
      *
      *   MISCELLANEOUS FIELDS
      *
       01  WS-SUB                        PIC 9(9) COMP.
       01  TRAN-BUF-LEN                  PIC 9(9) COMP.
       01  TRAN-BUFFER.
           05 TRAN-ARRAY OCCURS 100000 PIC X.
       01  LS-TRAN-BUF-PTR               USAGE POINTER.
       01  WS-ASCII-PTR                  USAGE POINTER.
       01  WS-EBCDIC-PTR                 USAGE POINTER.
      *
       01  ASCII-CODE-PAGE.
           05 FILLER PIC X(16) VALUE
               X'3A3B3C3E235B5D0708090A0B0C0D0E0F'.
           05 FILLER PIC X(16) VALUE
               X'101112131415161718191A1B1C1D1E1F'.
           05 FILLER PIC X(16) VALUE
               X'20212223242526272829 2A2B2C2D2E2F'.
           05 FILLER PIC X(16) VALUE
               X'303132333435363738393A3B3C3D3E3F'.
           05 FILLER PIC X(16) VALUE
               X'20414243444546474849 4A4B4C4D4E4F'.
           05 FILLER PIC X(16) VALUE
               X'505152535455565758595A5B2A5D5E5F'.
           05 FILLER PIC X(16) VALUE
               X'2D616263646566676869 6A6B6C6D6E6F'.
           05 FILLER PIC X(16) VALUE
               X'707172737475767778797A7B7C273D22'.
           05 FILLER PIC X(16) VALUE
               X'80616263646566676869 8A8B8C8D8E8F'.
           05 FILLER PIC X(16) VALUE
               X'906A6B6C6D6E6F7071729A9B9C9D9E9F'.
           05 FILLER PIC X(16) VALUE
               X'A0A1737475767778797AAAABACADAEAF'.
           05 FILLER PIC X(16) VALUE
               X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
           05 FILLER PIC X(16) VALUE
               X'C041424344454647 4849CACBCCCDCECF'.
           05 FILLER PIC X(16) VALUE
               X'D04A4B4C4D4E4F505152DADBDCDDDEDF'.
```

Figure 47 (Part 1 of 2). Copy Structure for EBCDIC-ASCII Translation

```
           05 FILLER PIC X(16) VALUE
               X'E0E1535455565758595AEAEBECEDEEEF'.
           05 FILLER PIC X(16) VALUE
               X'303132333435363738394FAFBFCFDFEFF'.
      *
      *   ASCII CODE TABLE
      *
       01 LS-ASCII-TABLE REDEFINES ASCII-CODE-PAGE.
          03  LS-ASCII-ENTRY         PIC X OCCURS 256
                INDEXED BY WS-ASCII-INDX.
      *
       01  EBCDIC-CODE-PAGE.
           05 FILLER PIC X(16) VALUE
               X'7A5E4C6E7B5F6A0708090A0B0C0D0E0F'.
           05 FILLER PIC X(16) VALUE
               X'101112131415161718191A1B1C1D1E1F'.
           05 FILLER PIC X(16) VALUE
               X'40212223246C26274D5D2A2B2C2D4B61'.
           05 FILLER PIC X(16) VALUE
               X'F0F1F2F3F4F5F6F7F8F93A3B3C3D3E3F'.
           05 FILLER PIC X(16) VALUE
               X'40C1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'.
           05 FILLER PIC X(16) VALUE
               X'D7D8D9E2E3E4E5E6E7E8E95B5C5D5E6D'.
           05 FILLER PIC X(16) VALUE
               X'60818283848586878889919293949596'.
           05 FILLER PIC X(16) VALUE
               X'979899A2A3A4A5A6A7A8A94D5D7D7E7F'.
           05 FILLER PIC X(16) VALUE
               X'8081828384858687888898A8B8C8D8E8F'.
           05 FILLER PIC X(16) VALUE
               X'909192939495969798999A9B9C9D9E9F'.
           05 FILLER PIC X(16) VALUE
               X'A0A1A2A3A4A5A6A7A8A9AAABACADAEAF'.
           05 FILLER PIC X(16) VALUE
               X'B0B1B2B3B4B5B6B7B8B9BABBBCBDBEBF'.
           05 FILLER PIC X(16) VALUE
               X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'.
           05 FILLER PIC X(16) VALUE
               X'D0D1D0D3D4D5D6D7D8D9DADBDCDDDEDF'.
           05 FILLER PIC X(16) VALUE
               X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'.
           05 FILLER PIC X(16) VALUE
               X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'.
      *
      *   EBCDIC CODE TABLE
      *
       01 LS-EBCDIC-TABLE REDEFINES EBCDIC-CODE-PAGE.
          03  LS-EBCDIC-ENTRY        PIC X OCCURS 256
                INDEXED BY WS-EBCDIC-INDX.
```

Figure 47 (Part 2 of 2). Copy Structure for EBCDIC-ASCII Translation

D.5 HTML in File Used by MVS File Browser Program

```
<!doctype html public "html2.0">
<html>
<head>
<title>Rob Macgregor's Home Page</title>
<meta name="abstract" content="This is 1st page of Rob Macgregor's Home Page">
<meta name="keywords" content="robmacg home page">
<meta name="owner"    content="robmacg@vnet.imb.com">
<meta name="review"   content="19960112">
<meta name="security" content="public">
</head>

<body>
<img src="header1.gif" alt="IBM"><br>
<h1>
Rob's Home Page
</h1>

<!-- end ibmhead---------------------------------------------------------- -->

<p>
<img src="servmini.gif">
I am an ITSO Assignee in Raleigh responsible for two areas:
<OL>
<LI>TME10 for AIX (and particularly NetView for AIX)
<LI>Network Security (Firewalls and WWW security)
</OL>
<P><HR>
<H2>Sample Code 'n Stuff</H2>
<P>When I produce redbooks with useful code samples, etc in them
I try to bundle them and place them on our
<A href="ftp://rsserver.itso.ral.ibm.com/pub">anonymous FTP server</A>.
You will find the samples under the /pub
directory, in subdirectories which match the book number.  So for example
the material from <CITE>Examples Using NetView for AIX V4</CITE>, SG24-4515
will be found in directory /pub/SG244515.
<p>
On the same server we also keep copies of
<A href="ftp://rsserver.itso.ral.ibm.com/pub/workshop_foils">workshop
foils</A> from the various workshops we run. You will find subdirectories
with names like "tokyo96" which contain the foils from the TME workshop
held in Makuhari, Japan during May 1996. You are free to make whatever use
of this material you like.
```

Figure 48 (Part 1 of 2). index.rob.html Used by MVS File Browser Program

```
<p><hr>
When I am not hewing wood and drawing water for IBM, I like to go sailing.
There is not much opportunity to do that here in Raleigh, although I have
a small Hobie Cat that I whizz about in. When I'm back in England I race
in my own 15ft Albacore dinghy, or on my brother's MG335:
<IMG src="flairII.gif">

<p> <hr>
<a href="http://w3.itso.ral.ibm.com/robmacg/mailme.html">
<img src="mail.gif"> </a>
If you have any comments on this home page or would like to send me mail.
<a href="http://w3.itso.ral.ibm.com/robmacg/mailme.html"> Mail me </a> anytime.
<p>
<a href="http://w3.itso.ral.ibm.com/ricardo/phones.html">
<img src="phone.gif"> </a>
Need to contact an assignee or a resident?
<a href="http://w3.itso.ral.ibm.com/ricardo/phones.html"> Here
</a> is the ITSO's phone directory.
<!-- end ibmbody------------------------------------------------------------ -->
<p> <hr>
<b>
[
<a href="http://www.ibm.com/">IBM home page</a> |
<a href="http://www.ibm.com/Orders/">Order</a> |
<a href="http://www.ibm.com/Search/">Search</a> |
<a href="http://www.ibm.com/Assist/">Contact IBM</a> |
<a href="http://www.ibm.com/Finding/">Help</a> |
<a href="http://www.ibm.com/copyright.html">(C)</a> |
<a href="http://www.ibm.com/trademarks.html">(TM)</a>
]
```

Figure 48 (Part 2 of 2). index.rob.html Used by MVS File Browser Program

Appendix E. MQSeries Definitions

This appendix contains MQSeries definitions for the MQGATE examples used in this document.

E.1 MQSeries OS/2 Application Objects

QLOCAL and QREMOTE definitions for the Web server are shown below.

```
DEFINE QLOCAL('Gateway.Reply.Queue') REPLACE +
       DESCR('Local queue for MQGATE interface') +
       DEFPSIST(YES)  +
       SHARE
DEFINE QREMOTE('WEB.CLIENT.QUEUE') REPLACE  +
       DESCR('Remote queue for WTPING/WTPONG') +
       DEFPSIST(YES)                +
       RNAME('WEB.CLIENT.QUEUE') +
       RQMNAME('CSQ2')
DEFINE QREMOTE('WEB.MGMT.QUEUE') REPLACE  +
       DESCR('Remote queue for Web File Browser') +
       DEFPSIST(YES)                +
       RNAME('WEB.MGMT.QUEUE') +
       RQMNAME('CSQ2')
```

Figure 49. Define OS/2 MQSeries Queue Objects

The connectivity objects were the same as used in B.2, "MQSeries OS/2 Connectivity Objects" on page 88.

E.2 MQSeries MVS Application Objects

QLOCAL and QMODEL definitions for the S/390 application servers are shown below.

```
DEFINE QLOCAL('WEB.CLIENT.QUEUE') REPLACE +
    DESCR('Local queue for WTPING/WTPONG') +
    DEFPSIST(YES)  +
    SHARE
DEFINE QLOCAL('WEB.MGMT.QUEUE') REPLACE +
    DESCR('Local queue for Web File Browser') +
    DEFPSIST(YES)  +
    SHARE
DEFINE REPLACE +
    QMODEL('WTPONG.REPLY.QUEUE')  +
    STGCLASS('SYSTEM') +
    DESCR('WT test application model')  +
    PUT(ENABLED) +
    DEFPSIST(NO) +
    MAXMSGL(4194304) +
    USAGE(NORMAL)    +
    SHARE            +
    DEFSOPT(SHARED)  +
    DEFTYPE(TEMPDYN) +
    GET(ENABLED)     +
    QSVCIEV(NONE)
```

Figure 50. Define MVS Objects

Appendix F. Downloading the Internet Connection Server for OS/2 and MQSeries Internet Gateway Support Pac

This appendix shows how the IBM Internet Connection Server for OS/2 and the MQSeries SupportPac MA81 for OS/2 was obtained.

All the following work was done on the Web server machine. In the following figures the download of the Internet Connection Server for OS/2 and the MQSeries SupportPac MA81 for OS/2 are shown. First we downloaded the Internet Connection Server for OS/2.

Step 1. Go to the IBM software home page http://www.software.ibm.com and select **Download Library**.

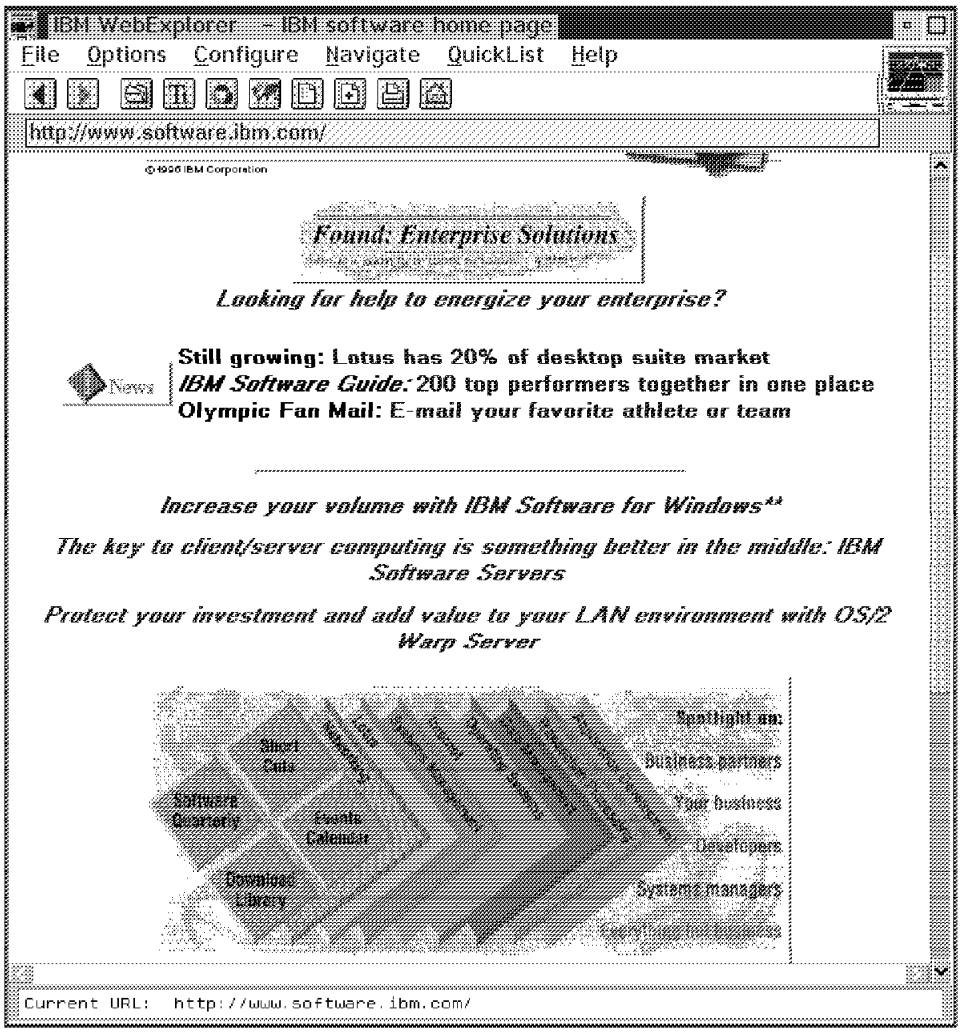


Figure 51. WWW Software Web Site

Step 2. Select **Beta/Evaluation Code of our products**.

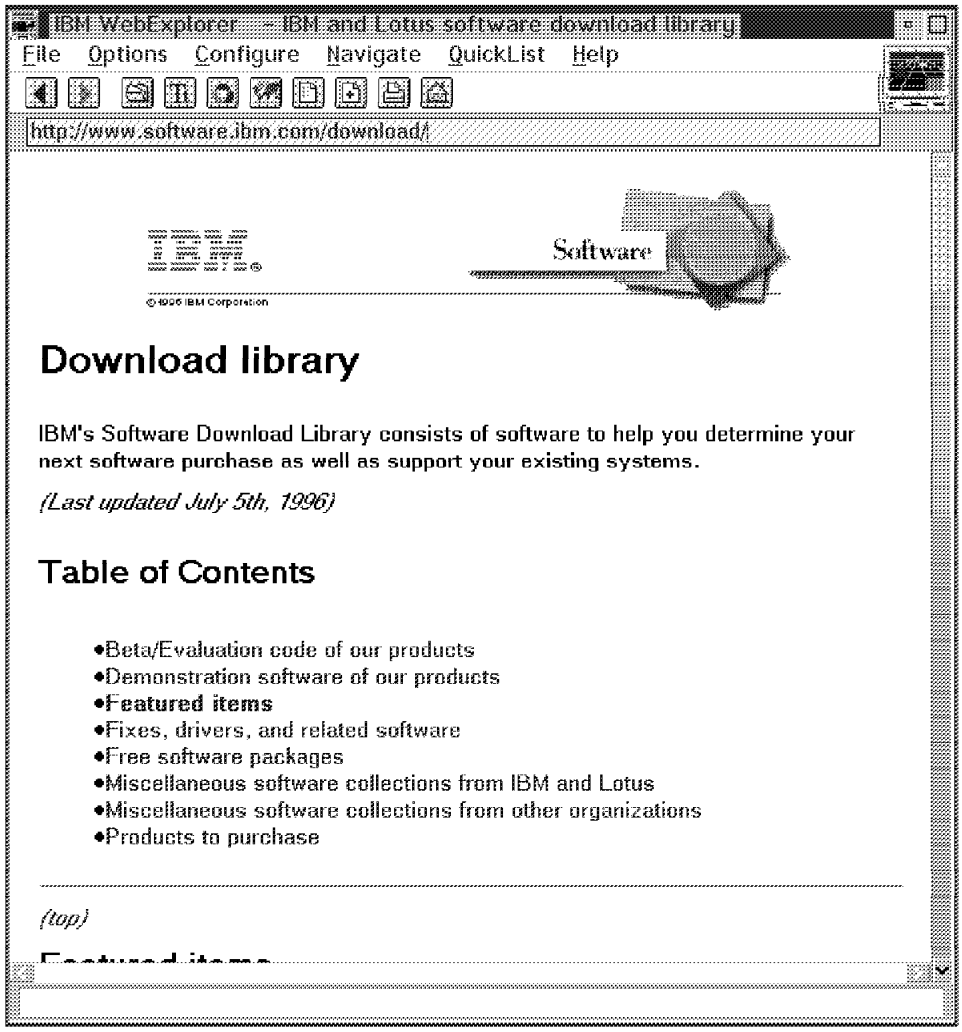


Figure 52. WWW Download

Step 3. Select **Internet Connection Servers**.

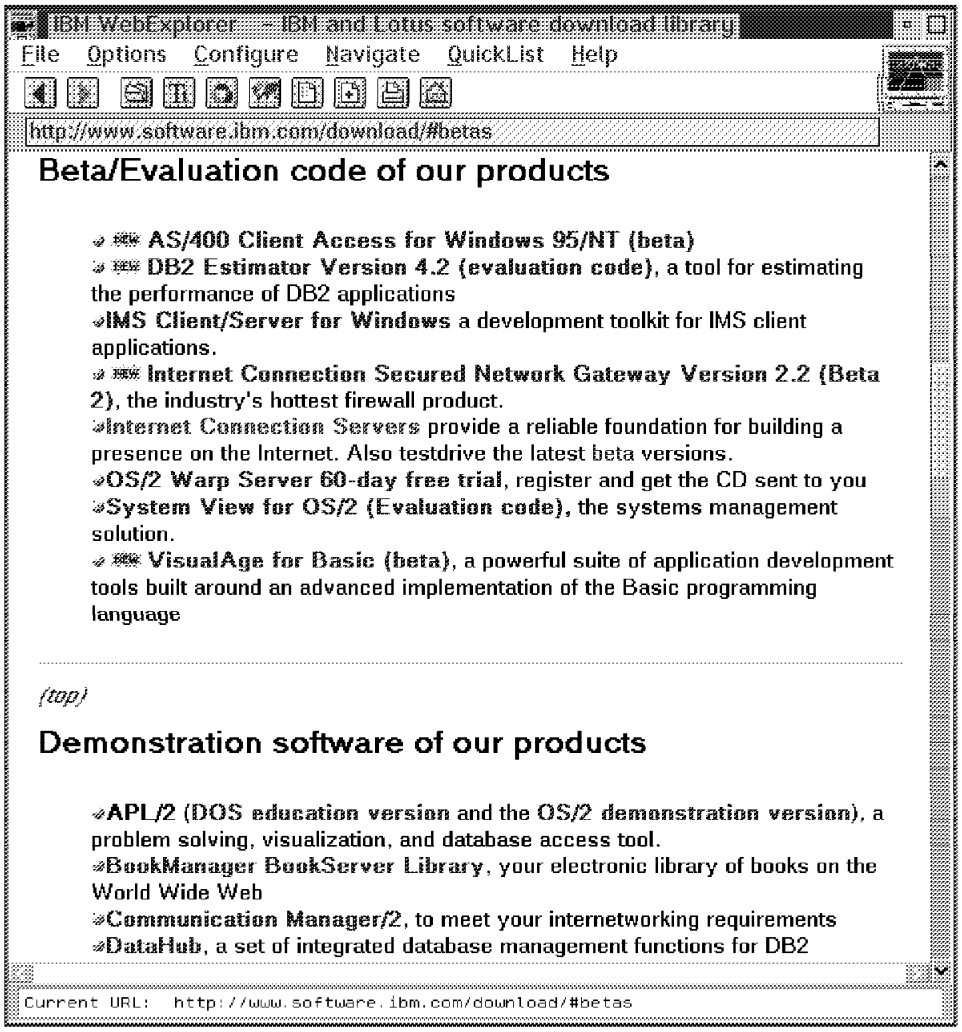


Figure 53. WWW Beta

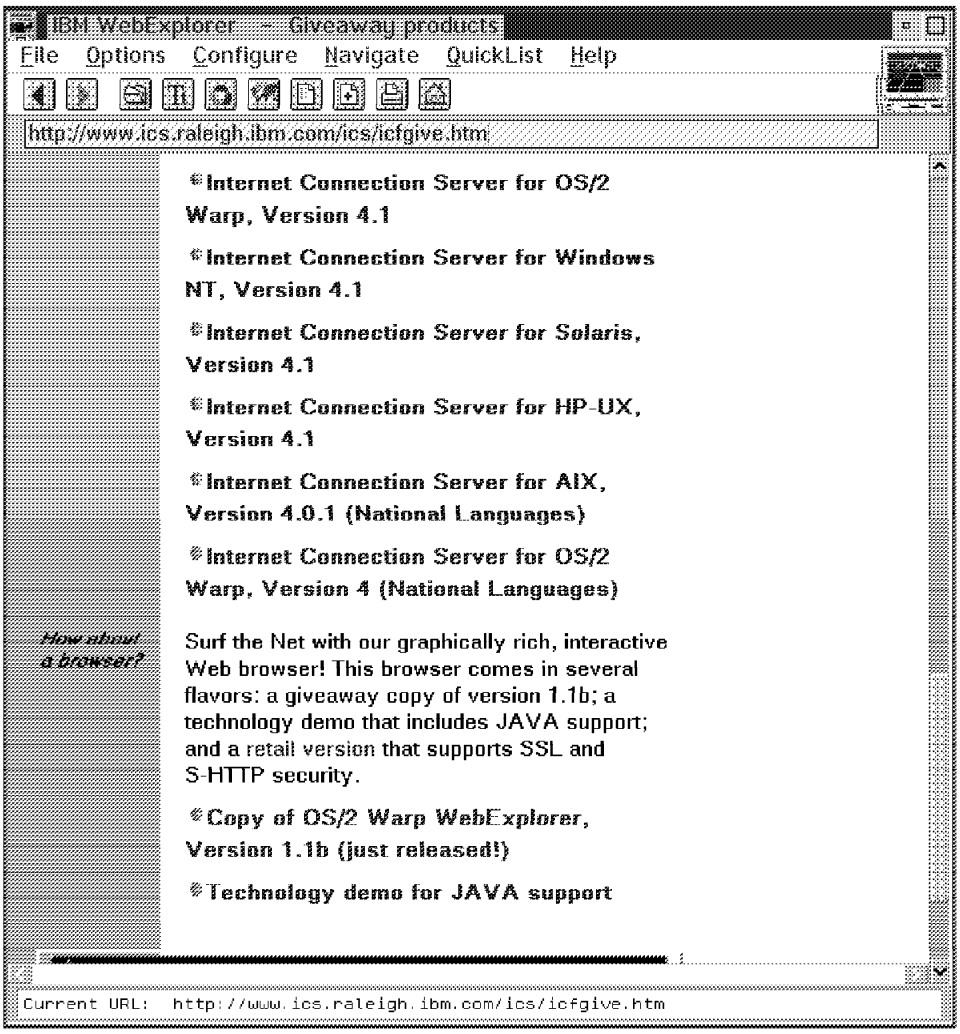


Figure 54. WWW ICS

From this window select **Internet Connection Server for OS/2 Warp 4.1** to download. Then select **download** in the Installation instructions. Download will transfer a ZIP file to your local machine from which installation can proceed.

Step 4. Select **MQSeries Internet Gateway** using http://www.hursley.ibm.com/cics/txppacs/txpsumm.html#int to download the package MA81.

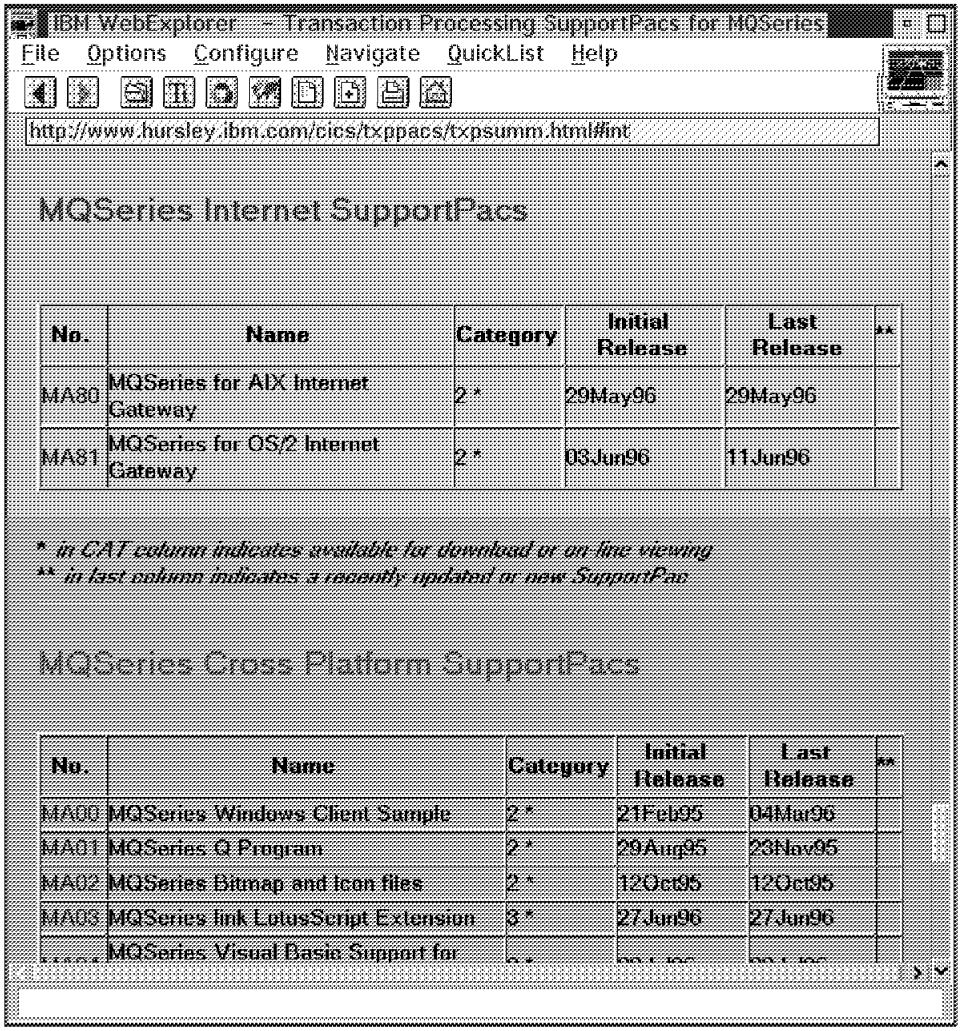


Figure 55. WWW MA81

From this window select download of the SupportPac MA81 for OS/2. Download will transfer a ZIP file to your local machine from which installation can proceed.

Installation of the Internet Connection Server for OS/2

After unzipping the downloaded file icsos41.zip, enter INSTALL (from the appropriate directory). This will install the product on the OS/2 machine. After rebooting the machine the Internet Connection Server is automatically started.

In order to configure the Web Server, start IBM WebExplorer on the machine as shown in the following figure. Then select **Configuration and Administration Forms**. A login screen is shown afterwards because the installation document is protected.

Please enter the userid webadmin and the password webibm.

Figure 56. WWW Login

For the configuration of the Web server go through the following steps:

- Select **Basic** and specify the required settings.

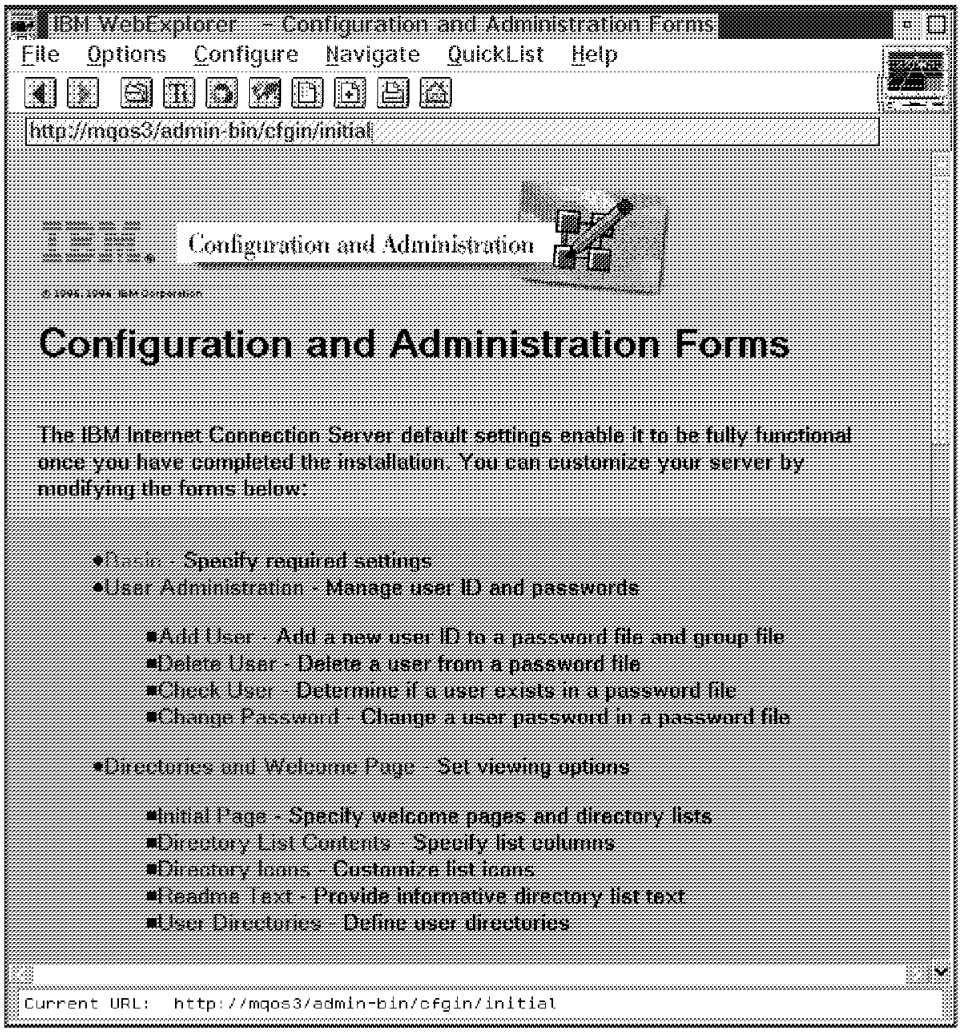


Figure 57. WWW Config

- Accept the Basic settings as seen on the screen.

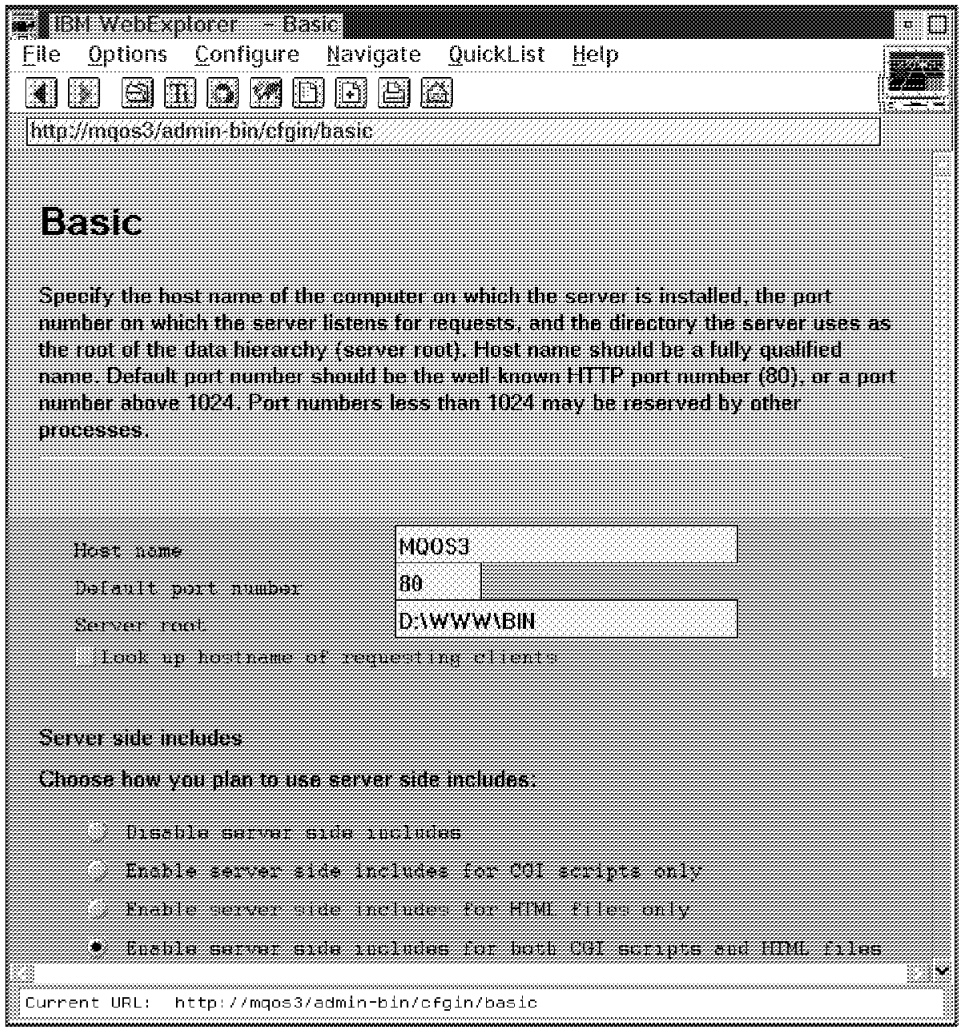


Figure 58. WWW Config Web Server

Now we proceed with the installation of the MQSeries Internet Gateway SupportPac. Again, this is done on the Web server machine.

First unzip the MA81.zip file from the root directory. Unzip creates a directory MQGATE and several subdirectories.

1. Change to subdirectory MQGATE\CGI-BIN and copy all files to the Web Server drive:\WWW\CGI-BIN directory.
2. Change to subdirectory MQGATE\HTDOCS and copy all files to the Web Server drive:\WWW\HTML directory.
3. Copy MQGATE\HTDOCS\IMAGES to drive:\WWW\HTML\IMAGES directory.

Now you can read the MQSeries Internet Gateway SupportPac User′s Guide from the WebExplorer by specifying the following URL: http://mqos3/MQGate.html

Appendix G. Special Notices

This publication is intended to help persons involved in installation and using MQSeries to understand how MQSeries can be used with an internet common gateway interface. The information in this publication is not intended as the specification of any programming interfaces that are provided by any products which are part of the MQSeries family or of the internet common gateway interface. See the PUBLICATIONS section of the IBM Programming Announcement for the MQSeries family products 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, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

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.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	IBM
MQ	MQSeries
MQSeries Three Tier	MVS/ESA
OS/2	S/390

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Microsoft, Windows, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Java and HotJava are trademarks of Sun Microsystems, Inc.

1-2-3, Lotus, Freelance, Freelance Graphics	Lotus Development Corporation

Other trademarks are trademarks of their respective companies.

Appendix H. Related Publications

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

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 203.

- *Examples of MQSeries Application Design and Systems Management*, SG24-4739 (available 1Q97)

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RISC System/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RISC System/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection (available soon)	SBOF-7250	SK2T-8042

H.3 Other Publications

These publications are also relevant as further information sources:

- *MQSeries Internet Gateway User's Guide* (included in SupportPac MA80/81)
- *MQSeries Application Programming Guide*, SC33-0807
- *MQSeries Application Programming Reference*, SC33-1673

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **ITSO4USA category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29570	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number _____
- Credit card number _____

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Index

Special Characters

A

B

C

D

H

I

M

R

S

T

W

Printed in U.S.A.