

WebSphere MQ Security in an Enterprise Environment

Cross-platform security

Secure Sockets Layer

Message security



Saida Davies
Peter Rhys-Jenkins
Hazel Fix
Mayumi Kawashima
John Scanlan
Steven Lane



International Technical Support Organization

**WebSphere MQ Security in an Enterprise
Environment**

May 2003

First Edition (May 2003)

This edition applies to Version 3, Release 5, Modification 0 of WebSphere MQ.

© Copyright International Business Machines Corporation 2003. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
Preface	xiii
The team that wrote this redbook	xiii
Become a published author	xvi
Comments welcome	xvii
Part 1. Enterprise security	1
Chapter 1. Project overview	3
1.1 Security is a process	5
1.2 Complexity	5
1.3 Some notes on terminology	6
1.3.1 Product names	6
1.3.2 Security terminology	6
1.4 Public key infrastructure (PKI)	7
Chapter 2. Planning	9
2.1 Planning methodology	10
2.1.1 Life cycle models	10
2.1.2 Security engineering	11
2.2 Perception	14
2.2.1 The definition of security	14
2.2.2 Myth 1: Security is a product	14
2.2.3 Myth 2: Security can be implemented without much planning	15
2.2.4 Myth 3: Security solutions can be considered in isolation	15
2.2.5 Myth 4: Security is a combination of protection methods only	16
2.3 Assets	16
2.4 Threat assessment	17
2.4.1 Threats against WebSphere MQ in an enterprise	17
2.4.2 Threat modeling	19
2.5 Risk assessment	22
2.5.1 Assessing loss expectancy	22
2.5.2 Other risk assessment methods	23
2.6 Policy development	23
2.6.1 Creating a security policy	23
2.6.2 Anatomy of a security policy	25
2.6.3 References for security policies and standards definitions	27

2.7 Policy implementation	27
2.7.1 What is a policy implementation document?	27
2.7.2 Elements of a policy implementation document	28
Chapter 3. Security technologies	31
3.1 Getting certificates.	32
3.2 Submit a request for a certificate.	32
3.2.1 Using the HTTP server solution (iKeyman)	32
3.2.2 Using a CA in test mode	33
3.2.3 Using Microsoft Windows 2000 certificate services	33
3.2.4 Using OpenSSL	33
3.2.5 Using Digital Certificate Manager (DCM)	33
3.2.6 Using RACF	33
3.3 Cryptographic co-processors	34
3.4 Algorithms	35
3.4.1 CipherSuites and CipherSpecs	35
3.4.2 HASH and MAC	36
3.4.3 Symmetric key encryption algorithms	37
3.5 Lightweight directory access protocol (LDAP).	38
3.5.1 Differences between directories and databases	38
3.6 KEYs and passwords	39
3.6.1 Password strength	39
3.7 SSL setup for WebSphere MQ	40
3.8 Planning for SSL	40
3.9 Preparing certificates.	41
3.9.1 Key repository	41
3.9.2 Defining the certificate type for your system	41
Chapter 4. Platform security.	45
4.1 z/OS security	46
4.1.1 Overview of z/OS security	46
4.1.2 z/OS Security Server.	48
4.1.3 Resource access control facility (RACF).	48
4.2 OS/400 security.	50
4.2.1 Object security	50
4.2.2 System security.	51
4.2.3 Physical security	51
4.2.4 Further reference	52
4.3 AIX security	52
4.3.1 User management and authorization	52
4.3.2 Network security	54
4.3.3 Trusted computing base (TCB)	55
4.3.4 Further reference	56

4.4	Windows 2000 security	56
4.4.1	Local logon process	57
4.4.2	Active directory	58
4.4.3	Network authentication	59
4.4.4	Group policies	60
4.4.5	Access control.	60
4.4.6	NTFS and the encrypted file system.	61
4.4.7	Public key infrastructure and certificate services	61
4.4.8	Auditing	62
4.5	WebSphere MQ security for z/OS	63
4.5.1	Overview	63
4.5.2	Switch profiles	65
4.5.3	WebSphere MQ for z/OS changes for V5.2	68
4.5.4	What has changed in WebSphere MQ for z/OS V5.2 security?	69
4.5.5	Connection security.	76
4.5.6	API security	76
4.5.7	Command and command resource security	79
4.5.8	User ID and reslevel	80
4.5.9	Security commands.	87
4.6	WebSphere MQ security for OS/400, AIX and Windows	88
4.6.1	WebSphere MQ security (OAM)	88
4.6.2	MQSeries administration wrapper (support pac MS0E)	91
4.7	WebSphere MQ security (alternate user authority)	92
4.8	WebSphere MQ channel security	93
4.8.1	Distributed queuing channel	93
4.8.2	WebSphere MQ client channel	94
4.9	WebSphere MQ security (channel exits).	95
Chapter 5. IBM Tivoli Access Manager for Business Integration		99
5.1	Revisiting the WebSphere MQ security issues	100
5.1.1	WebSphere MQ data flow	100
5.1.2	Native WebSphere MQ security issues.	101
5.2	IBM Tivoli Access Manager for Business Integration	107
5.2.1	Solving the issues with IBM Tivoli Access Manager for BI	107
5.3	A technical look at IBM Tivoli Access Manager for BI	116
5.3.1	Access manager and IBM Tivoli Access Manager for BI	116
5.3.2	IBM Tivoli Access Manager for BI Architecture	121
5.3.3	IBM Tivoli Access Manager for BI V4.1.	125
5.3.4	IBM Tivoli Access Manager for BI future evolution	126
5.4	Summary	127
5.4.1	Native WebSphere MQ Security vs. IBM Tivoli Access Manager for BI Security	127
5.4.2	When to use IBM Tivoli Access Manager for BI	127

Part 2. Securing WebSphere MQ	129
Chapter 6. Management issues	131
6.1 System validation	132
6.1.1 Problem definition	132
6.1.2 Attack simulations	132
6.2 Evolution	134
6.2.1 PKI certificate management	134
6.3 Management concerns summary	135
Chapter 7. Business scenario	137
7.1 Business to business	139
7.1.1 Environment summary	140
Chapter 8. Business scenario architecture	141
8.1 Overview	142
8.1.1 Dolphin Bank International system configuration	142
8.2 Hardware and software	146
8.2.1 zSeries	146
8.2.2 pSeries	148
8.2.3 xSeries	148
8.2.4 iSeries	149
8.3 Transactions from Windows through AIX to z/OS	149
8.3.1 Cluster class of service	149
8.3.2 Cluster object configuration	150
8.3.3 Queues	153
8.3.4 Applications	159
8.3.5 An example message flow step-through	159
8.4 File transfer from z/OS to OS/400	160
8.4.1 OS/400 configuration	161
Chapter 9. Business scenario security configuration	167
9.1 z/OS	168
9.2 AIX	168
9.3 Windows 2000	168
9.3.1 Client connections	168
9.3.2 The OAM settings	169
9.4 OS/400	170
9.4.1 Overview	170
9.4.2 Further reference	175
Chapter 10. Architectural vulnerabilities	177
10.1 CICS trigger monitor	178
10.1.1 Problem solution	178

10.1.2 Additional logic	179
10.2 Queue manager aliasing	180
10.3 What SSL does and does not provide	181
10.3.1 Why key length is important	183
10.3.2 Why is this important?	183
10.3.3 40 and 56 bit key summary	184
10.3.4 Using a key larger than 56 bits	184
10.3.5 CipherSpec recommendation	186
10.4 Attack types	186
10.4.1 Denial of service	186
10.4.2 Fraud	187
10.4.3 Password attacks	188
10.4.4 Access to certificates	189
10.4.5 Access to files	189
10.4.6 Access to binaries (executables)	190
10.4.7 Access to WebSphere MQ object definitions	190
10.4.8 LDAP access	190
10.4.9 Wireless LANs	190
10.5 Cryptographic co-processors	191
Chapter 11. Business scenario solution	195
11.1 Dolphin Bank International security policy	196
11.1.1 Policy for encrypting technology and certificate authority (CA)	196
11.1.2 Policy for WebSphere MQ on z/OS	197
11.1.3 Policy for WebSphere MQ on OS/400	199
11.1.4 Policy for WebSphere MQ on AIX	200
11.1.5 Policy for WebSphere MQ on Windows	200
11.2 WebSphere MQ objects for SSL	201
11.2.1 Queue manager	201
11.2.2 WebSphere MQ client	203
11.3 Security setup	205
11.3.1 Planning	205
11.3.2 Preparing a certificate on AIX	206
11.3.3 Preparing a certificate on Windows	216
11.3.4 Preparing a certificate on z/OS	224
11.4 WebSphere MQ setup	238
11.4.1 On z/OS	238
11.4.2 On OS/400	249
11.4.3 On AIX	255
11.4.4 On Windows	257
11.5 Cryptography: US export regulations	258
Appendix A. Good security practices	259

Naming conventions	260
Assign rights	260
OAM	260
Delete default items	260
Protect command server queues	261
Do not place everyone into the mqm group	261
Never have a blank MCAUSER	261
Change the CMDUSER on z/OS	261
Alias the DLQ	261
ALTUSER ID	262
Blank user IDs	262
Environment variables	262
Linear logging	262
Generic profiles	263
ISPF panel access	263
LU6.2 flowed user ID	263
Use aliases	263
CHINIT user IDs	263
Appendix B. Scripts, samples code and JCL	265
Script files by platform	266
z/OS	266
Object definitions	266
CLISTS for RACF definitions and access levels for initial setup	269
CLISTS for RACF definitions and access levels for business scenario	276
OS/400	303
Object definitions	303
AIX	304
Object definitions	304
Windows	308
Object definitions	308
JCL and code by platform	313
z/OS	313
OS/400	314
Sample code	315
Program source for SSL	315
Sample code on z/OS	317
Assembler program source	317
Assemble and link JCL	348
Execution JCL	349
CICS trigger monitor program source	350
z/OS assembler echo program	359
'C' message exit to restrict queue access	378

Appendix C. Additional information	389
Locating the Web material	389
How to configure HTTP Server Solution(iKeyman) certificate service . . .	389
How to configure Microsoft Windows 2000 certificate services	393
How to use OpenSSL	403
Glossary	407
Abbreviations and acronyms	411
Related publications	413
IBM Redbooks	413
Other resources	413
Referenced Web sites	413
How to get IBM Redbooks	414
IBM Redbooks collections.....	414
Index	415

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described 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:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

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

IBM®	NetVista™	SP2®
IBM eServer™	OS/390®	SP™
ibm.com®	OS/400®	SupportPac™
AIX 5L™	Parallel Sysplex®	Tivoli®
AIX®	pSeries™	WebSphere®
AS/400®	RACF®	xSeries™
CICS®	Redbooks (logo)™ 	z/OS™
Database 2™	Redbooks™	zSeries™
DB2®	RS/6000®	Domino™
ESCON®	S/390®	Lotus®
IMST™	SAA®	Notes®
iSeries™	Sequent®	Word Pro®
MQSeries®	SLC™	
MVS™	SP1®	

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Preface

In the first part of this IBM® Redbook, we provide an overview of the planning process that must precede any serious attempt to implement enterprise- wide security. We then describe a real world business scenario that incorporates multiple platforms:

- ▶ zSeries™ (z/900) CICS® (Customer Information Control System) and batch applications running under z/OS™ in a Parallel Sysplex®
- ▶ iSeries™ (AS/400®) based applications running under OS/400®
- ▶ pSeries™ (RS/6000®) based WebSphere® MQ queue managers acting as concentrators running under AIX®
- ▶ xSeries™ (Intel) based WebSphere MQ Server and client platforms running under Windows 2000

In the second part of the redbook, we document an existing WebSphere MQ environment. We explore security vulnerabilities in this environment. We then document how to apply WebSphere MQ technologies such as SSL (Secure Sockets Layer), ESMs (External Security Managers) and OAM (Object Authority Manager) to enhance enterprise security.

Finally, we close the book with a number of appendixes detailing common mistakes, platforms not covered by our chosen business scenario, source code, scripts and JCL that were used to create our “before” and “after” environments.

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, Hursley Center.

Saida Davies is a Project Leader for ITSO Hursley UK; she is a certified senior IT specialist and has 14 years of experience in IT. She has a degree in Computer Science and her background includes z/OS systems programming. Saida has extensive knowledge of IBM's z/OS operating system and a detailed working knowledge of both IBM and Independent Software Vendors' operating system software. In a customer facing role with IBM Global Services, she was engaged in the development of services for MQSeries® within the z/OS platform. This covers the architecture, scope, design, project management and implementation of the software on stand-alone systems or on systems in a Parallel Sysplex environment. One of her major projects was WebSphere MQ security review and

planning for a bank in Portugal in line with the British Standard code of practice for information security management. Additional experience includes project management, pre-sales support, training, system and subsystems migrations.

Peter Rhys-Jenkins is a senior architect for Candle Corporation. He designs complex messaging-based solutions for organizations world wide and has been involved in the success of many messaging middleware implementations. Peter has more than 20 years of experience building complex distributed systems on a wide variety of platforms, predominantly z/OS, Windows and UNIX based. He was a member of the team that created some of the IBM certifications for the WebSphere MQ family of products. Peter was the founder of AMSYS North America Inc., a messaging middleware consulting company that was acquired by Candle Corporation in 1996. He was educated in Wales at the Lewis School Pengam and at the University of Wales. Peter is an IBM Certified MQSeries Engineer, IBM Certified Solutions Expert – MQSeries, IBM Certified Developer – MQSeries, IBM Certified Solutions Expert - MQSeries Integrator, IBM Certified Specialist - MQSeries Integrator Version 2, IBM Certified Solutions Expert - MQSeries Integrator Version 2.

Hazel Fix is a Software Engineer who has worked for IBM UK for 15 years. She is one of the founding members of WebSphere MQ on z/OS and has worked on the product since 1991. She has been responsible for writing and maintaining the security function in WebSphere MQ for z/OS from the beginning. Hazel also spent a year seconded to the level 3 service group for WebSphere MQ on z/OS in the early days and still provides support for the group whenever required. In 2000, she was responsible for coordinating development of the Beta program for WebSphere MQ on z/OS V5.2 and ran the skills transfer education classes both in Hursley, UK and Raleigh, North Carolina, USA. In 2002, she coordinated the initial stages of the Beta program and the skills transfer education classes for WebSphere MQ for z/OS V5.3 in both Hursley, UK and Raleigh, North Carolina. Throughout her WebSphere MQ for z/OS career, she has also presented WebSphere MQ for z/OS security to customers and business partners.

Mayumi Kawashima is an Advisory IT Specialist for IBM Japan. She has worked with WebSphere MQ for seven years where she has developed WebSphere MQ and WebSphere MQ Integrator workshop material. She is responsible for WebSphere MQ skill transfer and technical support to IT specialists in IBM Japan. She is currently seconded to IBM Systems Engineering Co. Ltd. (ISE)., part of the IBM Advanced Technical Support Department, where she supports system design and implementation of WebSphere MQ solutions. In 1993, Mayumi transferred to ISE and first supported AIX (IBM version of UNIX system), Windows and DCE. Prior to that, she worked for the IBM Education Centre where she developed and taught AIX classes to IBM SEs and customers. She is an IBM certified specialist for MQSeries, an IBM solution expert for

MQSeries, a Red Hat certified Engineer for Linux and a Microsoft certified professional.

John Scanlan is an independent middleware specialist in the UK with five years commercial experience of IT in banking. His experience spans both retail and investment banking and he has been involved in successful WebSphere MQ production implementations that involve STP (Straight Through Processing). John has extensive experience with message routing and formatting, having worked for one of the major product vendors in this market segment. His areas of expertise include investment processing, file and message transfer products job scheduling and security. John has a degree in Mechanical Engineering from the University of London and is an IBM Certified MQSeries Specialist.

Steven Lane is an Information Security Consultant for Alphacourt Limited, a premier IBM business partner in the UK. He has worked with IBM WebSphere MQ for four years. He co-authored the IBM MQ250 training course entitled *WebSphere MQ: Designing and Architecting Clustering Solutions*. His areas of expertise include designing business integration solutions, enterprise networks and enterprise security. He is currently studying for a Master of Science degree in Information Technology at the University of Liverpool. He is an IBM certified specialist, a Microsoft certified systems engineer and a Novell certified network engineer.

The redbook team would like to thank the following people for their contributions to this project:

Robert Haimowitz, ITSO Raleigh, North Carolina for the Sysplex environment and support.

Morag Hughson, IBM Hursley for her invaluable support and help with RACF® and z/OS

Mike Horan, IBM Hursley for distributed SSL information

Andrew Banks, IBM Hursley for his help with complex clustering

Mike D Thomas, IBM Hursley for his support and help with OS/400

David Hay, IBM Hursley for his support and help with OS/400

Simon White, IBM Hursley for his support and help with AIX

Anthony O'Dowd, IBM Hursley for complex TCP/IP problem resolution

Richard Harran, IBM Hursley for his support with z/OS SSL

Amardeep Bhattal, IBM Hursley for his support and help with z/OS SSL

Walt Farrell IBM, Poughkeepsie for his support and help with RACF on z/OS

James Sweeney, IBM Poughkeepsie for his support and help with RACF on z/OS

Daniel Millwood, IBM Hursley for his support with z/OS Clustering.

David A Edwards, IBM Australia for his contribution on IBM Tivoli® Access Manager for Business Integration

Marc Verhiel, Candle Corporation Vancouver for the 'C' message exit program

Andrew Mattingly, Candle Corporation Sydney for "Common mistakes"

Susan E Leatherwood, Vanguard Corporation for the Cobol CKTI interface

Ian Bagley, Barclays Bank for OpenSSL guidance

The planning section uses "attack trees" to examine WebSphere MQ vulnerabilities, a technique that will be of value to any corporate security officer. We wish to thank Bruce Schneier of Counterpane Technologies for his permission to include this intuitive and practical technique in our redbook.

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

ibm.com/redbooks

- ▶ Send your comments in an Internet note to:

redbook@us.ibm.com

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization
IBM United Kingdom Ltd
MP135
Hursley, Hampshire S021 2JN



Part 1

Enterprise security

We have separated this redbook into two distinct parts, a theoretical part and a practical part.

This, the first part, is divided into five distinct chapters.

The overview, Chapter 1, “Project overview” on page 3, explains what we hope to achieve through the publication of this redbook. It looks briefly at security as a process rather than as a collection of loosely related technologies.

Chapter 2, “Planning” on page 9, examines security issues in the context of prevention, detection and reaction, explaining that a security system comprised solely of preventative measures is ineffective in dealing with today’s sophisticated cyber attacks. This planning section uses “attack trees” to examine WebSphere MQ vulnerabilities, a technique that will be of value to any corporate security officer.

Chapter 3, “Security technologies” on page 31, reviews most of the common cryptographic concepts and technologies that can be applied to enterprise security and attempts to explain these in plain English. This section hopefully can be used as a primer for those not wishing to go through several hard copy manuals and books.

Chapter 4, “Platform security” on page 45 looks at what technologies and procedures are available today that can be used to assist in securing the enterprise. Because our business scenario uses four platforms; z/OS (zSeries), OS/400 (iSeries) IBM AIX (pSeries) and Windows 2000 (xSeries), these are the platforms that we examine.

Chapter 5, “IBM Tivoli Access Manager for Business Integration” on page 99, the last chapter of this part of the redbook, examines risk and attempts to explain the risk management process and to quantify the cost-risk relationship to allow corporate management to decide what level of risk is acceptable in their environment and at what cost.

Security is a complex subject. We hope that this part of our book, devoid as it is of screen shots and sample code, will greatly aid the reader in understanding this complex subject. We hope that it will provide insight into the steps we have taken and documented in the second (practical) part of the book.



Project overview

WebSphere MQ is one of the foundation products of the WebSphere family. WebSphere MQ implementations all start small, yet over the years, the number and types of messages flowing through WebSphere MQ networks have grown for most enterprises. As message volumes have grown, the infrastructure supporting these messages has evolved.

This evolution has led to WebSphere MQ becoming the foundation for an enterprise wide, mission critical infrastructure, and supporting mission critical enterprise infrastructure transcends technology. That is why this book takes a hard look at security issues beyond technology.

Many enterprise customers, because they started out with small scale proof of concepts, did not fully implement WebSphere MQ security at an early stage, especially in distributed environments where the Object Authority Manager (OAM) was considered by many to be cumbersome and difficult to administer.

This book looks at an enterprise and documents the before and after configurations needed to take advantage of the recent functional improvements to WebSphere MQ, such as Public Key Infrastructure (PKI), Secure Sockets Layer (SSL) and the OAM.

If your enterprise has fully deployed the security capabilities of earlier versions of WebSphere MQ (when it was named MQSeries), then you may want to use this redbook as a refresher and skip to the sections that deal with new functionality, but we recommend that you read the entire book for a complete understanding.

Enterprise security is a vast, complex field of opportunities and challenges. In this book, we have attempted to lay the groundwork by describing some of procedures and documentation that need to be developed. By showing how SSL, coupled with external security managers, can close many security holes, we hope to lay the foundation upon which other Redbooks can layer and can include Certificate Revocation Lists (CRLs), Lightweight Data Access Protocol (LDAP), Online Certificate Status Protocol (OCSP) and performance characterization to allow the readers to make informed decisions as they trade performance against key strength. CRLs are the current working technology and could be important in closing an important gap to users' security.

Our justification for bypassing the use of the above for our business case scenario was based on our decision to manage a number of certificates at queue manager level and to ensure that these certificates were very manageable, allowing our systems administrator to remove a certificate from our key stores faster than a CRL could be updated.

Deciding to not support CRLs meant that we could also discount the need to set up and protect an LDAP directory, which is where CRLs are stored and used to validate certificates.

1.1 Security is a process

Implementing all of the technology and concepts in this redbook will not make your enterprise impregnable. It will make it more difficult to breach security, but a determined assault, whether for fraud or for reasons such as publicity and denial of service, will demand a process.

A security process involves protection, detection and action. Enabling Secure Sockets Layer V3 (SSL) on a WebSphere MQ channel is *prevention*, having an alert raised when a digital certificate proves that a message has been tampered with is *detection*, shutting down the channel and calling in the police is *action*. Protection by itself is pointless, and although it will deter many attackers, it will probably attract (more competent) others.

Without processes in place to detect and take remedial action to respond to such attacks, an enterprise lacks a complete solution.

Creating a whole solution starts with planning and this is described in the next chapter of this redbook. We then move from planning to examining existing infrastructure and making decisions as to what needs to be secured. The chapter then leads to deploying technology to implement those decisions, and to then look for vulnerabilities in the deployed environment, all the while deciding on the detection and action parts of the puzzle.

1.2 Complexity

Complexity, according to Bruce Schneier (author of *Secrets and Lies*, ISBN 0-471-25311-1) is the single biggest risk factor when deploying security infrastructures.

In our chosen business scenario shown in Figure 7-1 on page 138, we have taken a complex environment and tried to reduce it to manageable components by segregating the WebSphere MQ environment. Thus, we have a bottom tier composed of xSeries clients, a second tier of iSeries server instances, a third tier of pSeries concentrators, and a final tier composed of zSeries traditional mainframes. Despite this “less is more” paradigm, we have included complex WebSphere MQ technologies such as clustering technology on our Windows and AIX tier, and queue sharing groups on our parallel sysplex tier.

From a protection standpoint, each tier stands alone in our security environment. If you can secure the pieces, you can hopefully secure the whole.

Many security pundits state that you cannot secure an existing environment and that you have to start from scratch. We disagree, and as far as WebSphere MQ is concerned, we hope that this redbook will show you how.

1.3 Some notes on terminology

1.3.1 Product names

Many of the products noted in this document have been, or are in the process of being re-branded, perhaps leading the reader to be confused about the technology used in this redbook. For example, RACF (Resource Access Control Facility) in its latest incarnation is named z/OS Security Server, a name that is not in common usage at the time of the writing of this redbook.

Therefore, to prevent confusion, we have elected in this redbook to use the product names that we have all grown accustomed to over the years, and we shall continue to use terminology as it is used in our industry.

1.3.2 Security terminology

Much of the terminology associated with security is fairly new and abstract to the WebSphere MQ community. Rather than attempt to create our own definition of terms and perhaps confuse the reader, we have elected to reproduce some public domain explanations taken directly from the Public Key Infrastructure, x509 (PKIX) working group.

There are a number of terms used and misused throughout Public Key Infrastructure (PKI) cryptography, Public Management Infrastructure (PMI), Time Stamp and Data Certification literature. To limit confusion caused by some of those terms used throughout this document, we have produced the following list. Detail explanations are available in the glossary.

- ▶ Attribute Authority (AA)
- ▶ Attribute Certificate (AC)
- ▶ Certificate
- ▶ Certification Authority (CA)
- ▶ Certificate Policy (CP)
- ▶ Certification Practice Statement (CPS)
- ▶ End-entity
- ▶ Public Key Certificate (PKC)
- ▶ Public Key Infrastructure (PKI)

- ▶ Privilege Management Infrastructure (PMI)
- ▶ Registration Authority (RA)
- ▶ Relying party
- ▶ Root CA
- ▶ Subordinate CA
- ▶ Subject
- ▶ Time Stamp Authority (TSA)
- ▶ Top CA

1.4 Public key infrastructure (PKI)

A Public Key Infrastructure (PKI) is defined as the set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public key certificates based on public-key cryptography.

A PKI consists of five types of components:

- ▶ Certification Authorities (CAs) that issue and revoke certificates
- ▶ Organizational Registration Authorities that vouch for the binding between public keys and certificate holder identities and other attributes
- ▶ Certificate holders who are issued certificates and can sign digital documents and decrypt documents using private keys
- ▶ Clients that validate digital signatures and their certification paths from the known public key of a trusted CA and that encrypt documents using the public key from certificates holders
- ▶ Repositories that store and make available certificates and Certificate Revocation Lists (CRLs)



Planning

This chapter provides a methodology for designing large and complex enterprise WebSphere MQ security implementations.

In this chapter, the following topics are discussed:

- ▶ Life cycle models
- ▶ Security definitions
- ▶ Threat analysis
- ▶ Risk assessment
- ▶ Security policy development
- ▶ Security policy implementation

2.1 Planning methodology

This section covers life cycle models and security engineering.

2.1.1 Life cycle models

If a system design is approached in a non-structured way, the likely outcome will be a failure or a result that was not desired.

Large complex software systems require a disciplined and structured approach.

In the later part of the 1960s, the term *software engineering* was proposed and a structured approach to engineering software systems was developed.

Software engineering has been broken down into a number of processes. These processes are, basically:

- ▶ The specification of a system
- ▶ The design of a system
- ▶ The implementation of the design
- ▶ The validation of the design by testing methods
- ▶ The evolution of the system

These processes form the basis for a number of models. The first model is called the *waterfall* model.

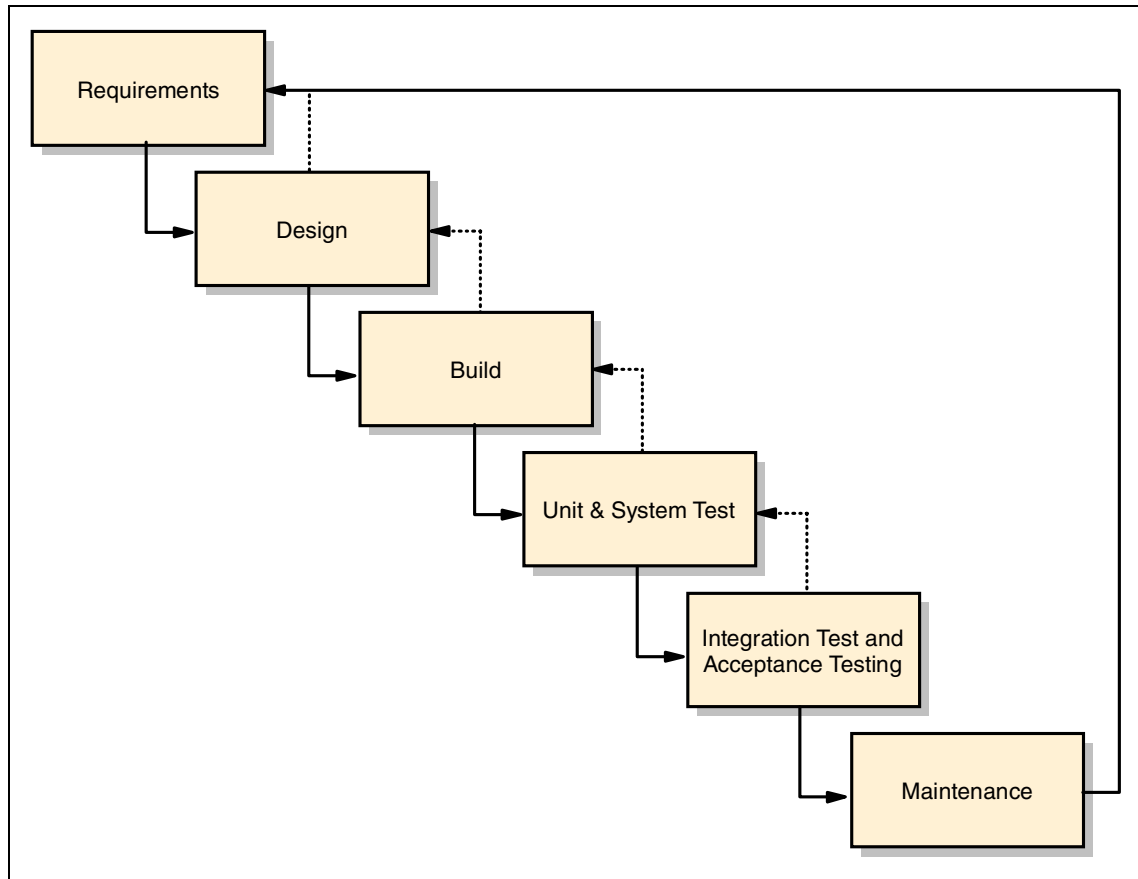


Figure 2-1 Waterfall life cycle

The waterfall model shown in Figure 2-1 was developed by Winston Royce in 1970 for the U.S. Air Force. The reference for publication is W.W. Royce, *Managing the Development of Large Software Systems: Concepts and Techniques*, in *Proceedings of the IEEE WESCON* (1970), pp 1-9.

To use this model, the requirements of the system must be known at the outset. In reality, this is rare since the requirements evolve as the system grows.

2.1.2 Security engineering

The process of designing and building a large enterprise security solution is complex. First, you need to understand the threats that you are trying to protect against, the adversaries that you potentially face, and the risk levels that are acceptable. It is impossible to build a functional secure system that cannot be

subverted. Therefore, you need to balance the level of risk that is acceptable to cost. In order to do this, the enterprise needs to have an understanding of the system and a clear set of goals. It needs to be sure that the final implemented technologies will satisfy those goals. Also, it needs to ensure that it reacts to new requirements usually emanating from new threats.

There are clear analogies in designing a security solution to software engineering. The term *security engineering* is widely used to describe the process of designing such a system. Security engineering is concerned with the methods used to design, test, build, implement and maintain the security of an environment as the risks and the requirements change. It is about building systems that remain dependable in the face of malign authorized and unauthorized users. It is about management of risk and calculated reduction of risk balanced with reward.

Security engineering can be expressed as a waterfall type model. Figure 2-2 on page 13 details a security engineering process model that is based on the “waterfall” life cycle model used in software engineering.

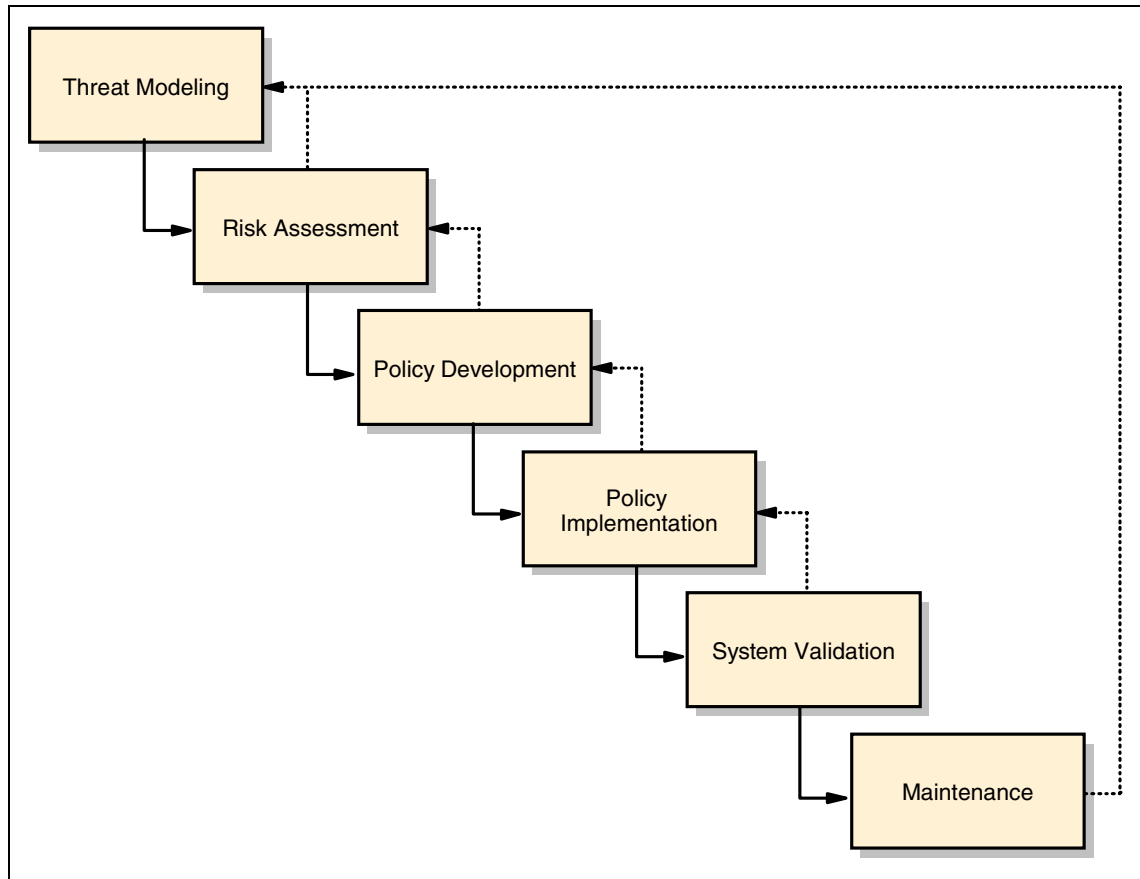


Figure 2-2 Security engineering process model based on the “waterfall” life cycle

In this chapter, we review threat modelling, risk assessment, policy development, and policy implementation as they apply to planning.

The practical application of such process modelling often requires a number of iterations before a system is complete. It is unusual for the design of a system to be complete after the first attempt.

2.2 Perception

This section describes different perceptions of security and common misconceptions about information security.

2.2.1 The definition of security

Before we take a further look at planning tasks, we need to dispel some common myths.

The word *security* encapsulates a number of ideas. For example, security could be taken by staff to mean that their jobs are secure.

Human resources could take staff security to mean ensuring that the staff in an organization are safe from harm.

Security guards on the front desk of a business are concerned with ensuring that the building is physically secure and unauthorized personnel are unable to enter.

Senior management may see security as in the financial security of the organization.

Different people throughout the organization will have different perceptions.

We are interested in information security. The process of securing a system is known as security engineering. However, you need to understand that information security is done within the framework of the overall business goals of balancing risk and reward.

Definition: Information security is the process of negating risk and balancing the cost of a security solution implementation and subsequent support with the business rewards.

2.2.2 Myth 1: Security is a product

A common misconception of information security is that you can buy it. Perhaps you can demonstrate due diligence by installing and paying for a firewall, but you cannot say that implementing a firewall makes a company secure.

Security is ultimately reduced if staff make statements such as “You are secure because you have implemented firewalls /password logons/ SSL”. It is important to understand that information security is a process and not a product.

2.2.3 Myth 2: Security can be implemented without much planning

If you do not plan with a top down approach, assessing the risks and developing a policy, you cannot develop a security solution. You must have a security policy. Unless you have a cohesive security policy that is implemented system wide and regularly reviewed, security is pointless. You may have implemented the best technology available at great expense but you cannot define what it is supposed to do without a policy.

2.2.4 Myth 3: Security solutions can be considered in isolation

A policy must address the system as a whole. The perception that a single technology can solve security goals is wrong. Security in a system is like a chain. If one link is made of paper, the whole chain is insecure. The chain is only as good as the strength of each link and a weak link means the chain is not fit for the purpose.

Computing is no different and you need to ensure that all links in the chain are secure.

You need to consider the whole system and not just its parts. A policy for security can be implemented on an operating system such as UNIX. When this is connected to the network, you need to ensure that the network policy is implemented and suitable.

The combination of UNIX and the network forms a system.

Therefore, the security policy should look at the system as a whole. This will probably materialize as two policies interlocked as shown in Figure 2-5 on page 25.

Add a new component such as WebSphere MQ to the server and you have a new system. A security policy for WebSphere MQ should be developed. However, it is possible that the platform security policy and the network need to be revised.

The system as a whole should be considered once again to ensure that the interlocking set of policies constitute a comprehensive solution.

It is wrong to perceive that because a system was considered secure before WebSphere MQ, adding security to WebSphere MQ will secure the overall system or achieve the security goals. Taking this approach leads to insecurities.

As the system becomes larger, the probability that a weak link exists increases. Policies and solutions should reinforce each other in an interlocking manner.

2.2.5 Myth 4: Security is a combination of protection methods only

Finally, the perception of information security in most organizations is that of protection. Many organizations give no thought to detection and reaction. These are just as important, if not more important than the protection methods.

If you implement a WebSphere MQ attack detection method, you must have a plan as to how you are to deal with the attack. You should have practiced the reaction method in much the same way as a fire drill.

Protection, detection and reaction are all part of a security solution.

2.3 Assets

Before identifying threats to your system you need to identify:

1. What you need to protect. For example, this could be your corporate assets.
2. Who your adversaries are and what are their motivations.
3. What resources your adversaries have available to them.

An enterprise may decide to classify its assets into different groups. This classification system often helps define different criteria to different parts of a system. For example, you may classify your assets using the following labels.

- ▶ Secret
- ▶ Company confidential
- ▶ Company sensitive
- ▶ Restricted
- ▶ Unclassified

Each of these labels would need to be defined in the context of the enterprise.

Having ascertained what to protect, you should consider your adversaries and their motivations. It is important to look inside and outside an enterprise and realize that many security breaches occur simply as operator errors.

2.4 Threat assessment

This section discusses threats against WebSphere MQ in an enterprise and threat modelling.

2.4.1 Threats against WebSphere MQ in an enterprise

This section explains some common threats to WebSphere MQ security.

Sniffing: Computers with access to a network can record the traffic flowing through it. If data or commands are sent unencrypted as WebSphere MQ messages, it is easy for unauthorized people to passively eavesdrop. The keyword here is *passive*. Sniffing is an activity that leaves no trace in a network log.

Sniffing is a threat to confidentiality, but if user IDs and passwords are sniffed, the threat becomes more serious because the attacker could then impersonate a legitimate user. It may be possible for an attacker to also use sniffed packets off the network and replay them. For example, if an MQ message is an instruction to make a payment, then it is possible that the attacker can replay this message a number of times. This is demonstrated in Chapter 10.1, “CICS trigger monitor” on page 178.

Impersonation: The attacker tricks a security system, passing as an authorized user. There are three possible levels of impersonation. Firstly, the attacker may be able to impersonate the WebSphere MQ administrator. Secondly, the attacker may be able to impersonate a valid queue manager and receive messages. Thirdly, an attacker may be able to impersonate a sending queue manager and send messages.

For example, the attacker could impersonate a valid queue manager and pass messages. This may be easy for the attacker to do, given that most queue managers in a distributed environment have the administration ID set to mqm and that queue managers do not authenticate by default.

The attacker may be able to steal a valid user ID and passwords by recording network traffic when a WebSphere MQ administrator signs on. If a WebSphere MQ message is not digitally signed or encoded with a weak CipherSpec, an attacker can modify or enter completely new data or commands.

Impersonation can be a threat to all three goals of computer security.

Decryption: If WebSphere MQ messages are sent over a public network, attackers can often easily obtain the encrypted data. If the encryption is weak,

the attackers can decrypt the data in a fairly short time. Decryption is a threat to confidentiality.

Flooding: If an attacker sends large amounts of data, it will fill the network bandwidth. If the attacker has gained access to a WebSphere MQ queue manager, the server may become overutilized, preventing access to other users or greatly affecting performance. An attacker may also be able to flood downstream queue managers by using the communications between queue managers. Flooding is a threat to availability.

Technology or application weakness: The TCP/IP protocol, some of its applications, and some operating systems have inherent security shortcomings, sometimes due to the objectives of their original design (openness, easy communication between computers and applications).

Security holes in the underlying technologies such as platforms and network protocols affect the security of WebSphere MQ.

Company-developed applications such as WebSphere MQ adapters or software purchased from vendors may have security weaknesses that attackers can exploit.

The degree of the damage depends on the nature of the problem. The most common damage is for a system to be shut down.

The problem could be more serious, allowing the attackers access to data that they can alter or use to their advantage. Technology and application weaknesses exploited by malicious attackers are threats to all goals of WebSphere MQ security.

To protect an enterprise, you must keep up to date with the vendors' security updates and rely on the providers with good reputations to pay attention to security.

If an enterprise develops its own applications to run on hosts, security must always be at the top of the design goals, whether they interface into WebSphere MQ or not.

2.4.2 Threat modeling

This section covers methods used to model threats.

Using Fault Tree Analysis and threat trees

There are a number of formal methods that can be used to model threats. A common approach to modelling a threat is borrowed from the world of safety critical systems.

In safety critical systems analysis and design, a method called Fault Tree Analysis (FTA) is used. This method models the failures that cascade to a critical failure. The root node is a catastrophic failure of some kind. This method can be extended to the security engineering world to form a threat tree. In a threat tree, the root node is some undesirable outcome, such as Denial Of Service (DOS).

From the root emanate branches to nodes. Each of the nodes contains a description of the conditions that need to be met to achieve the DOS.

The nodes can be OR nodes or AND nodes. Therefore, one or more conditions must be met to achieve the DOS attack.

From each node, a branch (or branches) emanates if other conditions need to be met in order to achieve the node. The subnodes are modeled in the same method using OR or AND. The tree is extended until each node and subnode is mapped out.

The method then assigns probabilities to each of the nodes, taking different adversaries into account, from the casual hacker to the well funded and determined agency. Using this method, you can clearly see where the risks lie, what the most probable attack method is, and where an enterprise should apply protection methods to reduce risk.

Using attack trees for threat modeling

An interesting development of this method is the attack tree as invented by Bruce Schneier of Counterpane Internet Security Inc.

Tip: For detailed information on this method, go to:

<http://www.counterpane.com/attacktrees-ddj-ft.html>

or refer to the book by Schneier, *Secrets and Lies, Digital Security in a Network World*, John Wiley & Sons Inc; 2000, ISBN 0471253111

Attack trees are a very similar to threat trees. The root node is some undesirable outcome. The major difference between threat tree analysis and attack tree

analysis are the values that are assigned to nodes and the calculations that you make.

You can make quantitative comparisons of attacks. For example, you can evaluate how complex an attack is in terms of specialist knowledge, specialist equipment, and total cost to the attacker, and then assign some realistic probability that an attack is possible.

The aim here is to give an overview of the method in a simple manner. We will not explain this method fully. In the references given above, there is a great deal more information on this topic and the references should be read before attempting to use this system. There is a real world example of an attack tree on Pretty Good Policy (PGP) in the references.

As an example, consider a very simple system. The system consists of one server. The server is an xSeries Windows 2000 server that we will call XSSERV running WebSphere MQ V5.3 queue manager. The queue manager is called MQW2K. The server is on an internal Local Area Network (LAN) running TCP/IP. The TCP/IP network is firewalled from the Internet. A UDB/DB2® database runs on the Windows 2000 server.

Users run an application on their local PC that connects to the MQW2K queue manager using the WebSphere MQ client. The PCs are in a workgroup and do not participate in Windows 2000 domain security. The XSSERV is also a stand-alone server that is not in a Windows 2000 domain. The client channel that the clients connect to has an MCAUSER ID set to mquser. This user ID has been defined as a local user on the XSSERV and has the authority to connect to the queue manager, put to the input queue and get from the reply queue. The queue manager has been installed with all the default values and with all default objects in place. The command server is running and the administrator has defined a server connection channel to allow remote administration with an MCAUSER ID set to an administration ID.

The client application puts a message to an input queue and an adapter takes the message, queries the database and puts the response to the ReplyToQ.

The undesirable outcome is to cause a denial of service attack (DOS) on the server. Therefore, we will set this as the root note.

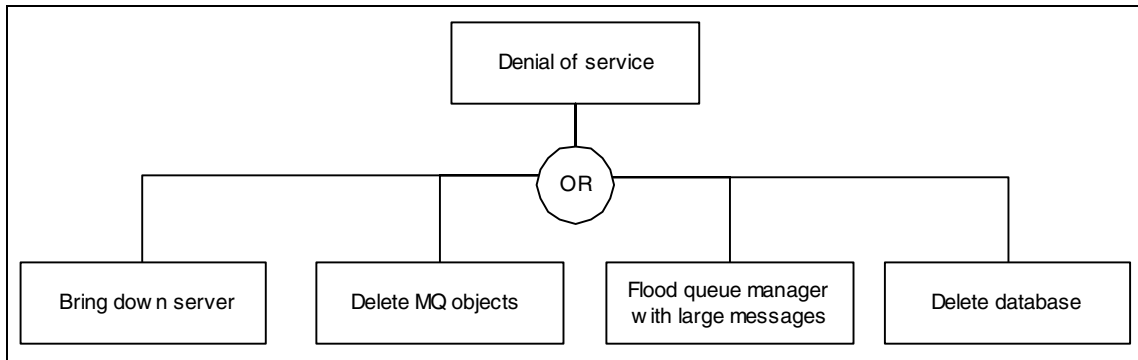


Figure 2-3 Root attack tree for a denial of service attack

Each of the subnodes in this tree then becomes a goal in itself. The subnodes in the subtree are either ORed or they are ANDed in a boolean manner.

For this simple example, we will develop one of the subnodes. That subnode is to delete MQ objects. Since this queue manager has been installed as default, it is quite vulnerable to attack.

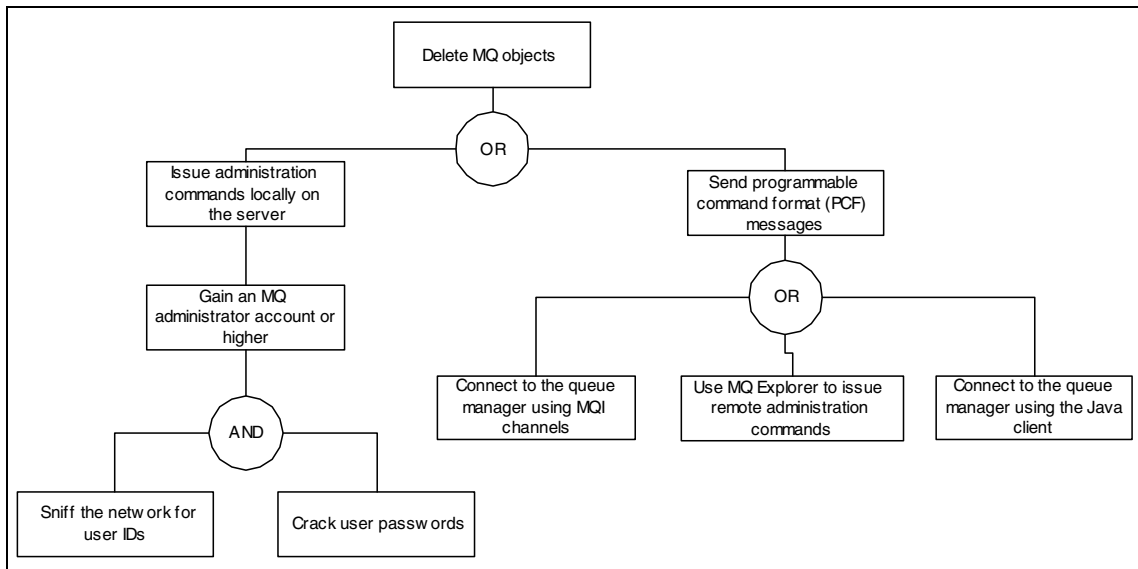


Figure 2-4 Delete MQ objects attack tree subnode

In the delete MQ objects subtree, you can see some of the possible ways that this can be done. In a full attack tree, we would have endeavored to have evaluated all the possible ways.

The next step would be to evaluate each of the subnodes to see if they are possible, if they require specialist knowledge, how much they will cost to conduct and so forth. The idea is to build a picture of the system from a threat point of view. The more evaluation is made of the attack tree, the more an understanding of the system will be gained.

2.5 Risk assessment

The following topics are discussed in this section: assessing loss expectancy and other risk assessment methods.

2.5.1 Assessing loss expectancy

It may be possible to estimate the cost of an event occurring and associate some figures to the risks identified in the threat model.

A common way of completing this task is to use a method called Annual Loss Expectancy (ALE). The ALE is calculated by listing all the threats identified in the attack tree, estimating the amount that an incident would cost the business, and multiplying it by the number of incidents per annum. For example, the cost to the business if a WebSphere MQ system was subjected to a Denial Of Service attack may be calculated as follows.

Item	Cost
Loss of business due to down time	\$100,000
Cost of consultants to bring system back online	\$10,000
Cost of investigation to discover source of attack	\$10,000
Indirect cost (for example loss of confidence)	\$100,000

The total cost of Denial Of Service attack to the business could be an estimated \$220,000. However, the number of incidents per year may be only 0.05. Therefore, the ALE is calculated by multiplying 0.05 by \$220,000 to give an ALE of \$11,000.

Please note that these figures are an example only and are not a representation of the real risk of a DOS attack.

While the ALE is clearly an estimate, it is more accurate if it is based on real figures. However, where real figures are not available it really comes down to the experience of the practitioner and the probability of an attack gained from a

knowledge of the attack tree. The harder an attack, the lower the probability that this attack will occur.

The ALE is useful for assessing investment in technology to provide protection and detection. In the given example, it would be unreasonable for the business to spend \$100,000 on technology to remove the Denial Of Service attack risk . However, the business may get a good return on investment if the risk could be negated with a spend of \$5,000.

2.5.2 Other risk assessment methods

Restriction: The methods given below are beyond the scope of this document to detail. However, considerable information exists on the Internet.

Other formal methods of risk analysis exist that are relevant to security also exist. Two examples are the “Central Computer and Telecommunications Agency Risk Analysis and Management Method” (CRAMM) developed in 1985 by the UK government and “Consultative, Objective and Bi-functional Risk Analysis” (COBRA), used internationally.

The scope of these methods falls outside this document but may be appropriate in your organization.

2.6 Policy development

A security policy is essential for developing a security solution. It is simply not acceptable to implement a point security solution and call the system secure. You should identify the threats against which you wish to defend through threat and risk analysis.

2.6.1 Creating a security policy

“Security policy” is one of those phrases that has come to mean a number of different things to different organizations. We believe that a security policy should state what the protection, detection and reaction methods are designed to achieve. It should be driven by the risk assessment activities of threat modelling and risk analysis. It should be clear, concise and avoid open-ended phrases that can be interpreted in multiple manners.

The more clear and concise a policy is, the more likely it is that it will be followed. The more open-ended the policy is, the more it will be interpreted and likely to be misinterpreted.

A good security policy should not state “how” a system should be protected but “what” should be protected and why. In the security policy, you should directly address the threats.

A good security policy will take the system as a whole. By *system* in a WebSphere MQ enterprise, we mean the whole system from end to end. For example, in a three-tiered WebSphere MQ environment with a Web front tier, leading to a concentrator middle tier, and finally to a back-end tier such as a database, you would be concerned with the system from end to end. You would be concerned with the policy for Web application servers such as WebSphere Application Server, the platform on which the server runs, the policy for the network, the policy for any software running on the platform such as WebSphere MQ, and so on.

In a system that includes a Business to Business (B2B) link, you may even have to consider the security of the business partner. The policy for this system will be part of the entire IT security policy, and reference other documents, or layers on top of other policies such as platform policies. The important point to note is that the existing policies should be reviewed to ensure that they fit with the overall system change.

There is no hard and fast way of writing a policy document. Organizations often have internal standards for documentation of this nature. It is unusual in larger organizations for policies not to exist. However, if there are no policies within your organization, you may need to develop a document structure from scratch.

The policy should be written in simple language, avoiding technical jargon where possible, so as to be understood at both board level and technical level.

The technical details of the policy, the implementation details, can be left until the next stage. It is important to get a full agreement throughout the organization. Without board level support, the budget for the implementation will not be available or not justifiable.

Without agreement from the administrator of a system, a policy will not be implemented. Additionally if the policy is too complex it will not be implemented. Everyone at all levels must understand what you are attempting to achieve and why.

A policy must also prepare a response to an attack or an accident that could compromise security, for example:

- ▶ Keep up-to-date lists of people and organizations to contact in case of a security emergency. Include the names of the persons in the enterprise expected to make those calls.

- ▶ Make a list of the most likely attacks to which the network is vulnerable and consider what you should do when they happen, for example:
 - Should you immediately disconnect your network from the Internet?
 - How can you track the attacker?
 - What supporting documentation is required?
 - Who must be informed?
 - If contacted by the media, what should you tell them?
- ▶ Have drills of rehearsals to verify that your people and organization react according to plan.

Important: You may not be able to prevent an attack, but you can avoid being unprepared for it.

2.6.2 Anatomy of a security policy

A WebSphere MQ policy document will be part of the IT security policy, which in turn will be part of the corporate security policy.

WebSphere MQ will not be secure if the other layers are not secure. One way of organizing a security policy document is to think of it as concentric rings. Figure 2-5 gives an example of this structure.

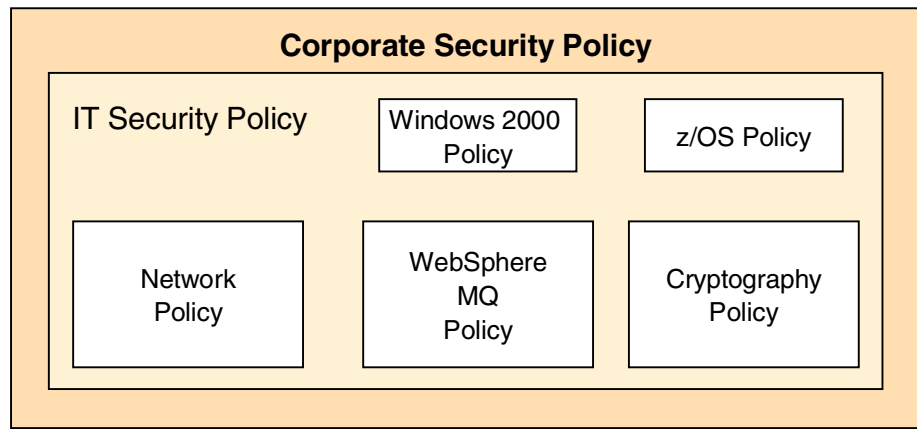


Figure 2-5 Organizing WebSphere MQ security policy within the framework of an organization

Note: It is beyond the scope of this redbook to provide detailed information on overall security concepts, policies, and processes. An excellent starting point for this subject is RFC 2196, *Site Security Handbook*.

Some examples of security policy components are as follows.

- ▶ Guidelines on required and preferred security features of new products that the company purchases
- ▶ Privacy policy dealing with electronic mail, keystrokes recording, files stored on company's media, and other use of company's resources
- ▶ Decision regarding which messages must be displayed warning users that they might be monitored and informing them that only authorized access is permitted
- ▶ An Acceptable Use Policy(AUP) that clearly defines the purposes for which the company's systems and networks may be used
- ▶ Responsibilities of users, IT staff, and management, and how each of them should handle a security incident
- ▶ Which connections are allowed to external networks and systems
- ▶ What services are permitted from the internal network to the Internet, who is authorized to access those services, and what restrictions apply
- ▶ Same as above but from the Internet to the company's network
- ▶ How the configuration of systems and networks may be changed and who may change them all
- ▶ Who is allowed to access what systems and in which ways they may access those systems
- ▶ How to authenticate users, passwords requirements; local and remote user authentication guidelines
- ▶ Availability of resources, how to achieve the desired level of availability and performance and how to measure the service level, how to monitor for deviations from the normal or expected values, and what to do when an availability or performance anomaly is detected
- ▶ Who is authorized to perform maintenance of systems and networks, and especially which type of remote maintenance is allowed; how the authorized maintenance personnel will prove their identity
- ▶ How to report policy violations, including contact information

- ▶ How to handle queries about security incidents and requests for confidential information
- ▶ Cross reference to security procedures and other documents (policies, laws, government regulations)

2.6.3 References for security policies and standards definitions

It is important that your implementation follow your company's security policies and security standards.

You can find good guidelines for developing computer security policies and procedures for sites that are connected to Internet in the following documents and Web sites:

- ▶ RFC 2196, *The Site Security Handbook*:
<http://www.rfc-editor.org/rfc/rfc2196.txt>
<http://www.faqs.org/rfcs/rfc2196.html>
- ▶ National Institute of Standards and Technology:
<http://cs-www.ncsl.nist.gov/policies/index.html>
- ▶ Center for Information Technology/Security:
http://irm.cit.nih.gov/security/sec_policy.html
- ▶ A number of off-the-shelf policies exist on the Internet, for example at:
<http://www.sans.org/newlook/resources/policies/policies.htm>.

Although there are no published WebSphere MQ policies at this URL, it is educational to study the format of the example policies. However, you are urged to spend time to develop a policy of your own. In doing so, you will gain a better understanding of your system and the end policy will be much better.

- ▶ Some interesting additional background reading on policy development can be obtained at:
http://rr.sans.org/policy/policy_list.php.

2.7 Policy implementation

This section discusses aspects of policy implementation.

2.7.1 What is a policy implementation document?

Some of the best policies developed are never implemented properly. A security policy is not sufficient. You should refine your security policy document into a security implementation document or set of documents.

Whereas a security policy seeks to define what is to be protected and why, an implementation document should state how.

The document should detail without ambiguity the technology to be implemented to achieve the policy goals. It should be a detailed guide to implementing the policy.

For example, a policy for WebSphere MQ might state that messages must not be read or altered while on the network. The implementation document would state that SSL will be implemented to achieve this aim. Furthermore the implementation document should detail a step-by-step guide to implementing this technology so that steps are always repeatable in the same way.

2.7.2 Elements of a policy implementation document

The actual format of the policy implementation document will vary from organization to organization. One of the best ways to write this document is to take specific goals from the policy document and address each goal with a technical solution. The solution should be evaluated for its suitability. It should achieve the policy goal fully and the implementation must be justified.

Some of the suggested elements of policy implementation documents are as follows:

- ▶ Policy goals- particular to the component of the system that is to be secured
- ▶ Roles and responsibilities - who is responsible for doing what
- ▶ Technology - what solutions have been selected and what the selection criteria is
- ▶ Step-by-step guide - ensuring that the localized implementation is repeatable in an exact fashion (this may simply be a pointer to a script for automatic setup)
- ▶ Maintenance - how the implementation document and technologies are to be maintained and how often

Note: System validation, maintenance and management are covered in Chapter 6, “Management issues” on page 131.

Tip: For further information on security engineering, you are advised to read the excellent book *Security Engineering* by Anderson. Information on the book is available at <http://www.cl.cam.ac.uk/~rja14/book.html>

Dr. Ross Anderson has written a number of Information Security papers that provide excellent reading. Go to <http://www.ross-anderson.com> for more information.



Security technologies

This chapter takes a look at some of the technologies used to secure WebSphere MQ infrastructures. It focuses on cryptography and products, such as SSL, that implement cryptographic algorithms.

3.1 Getting certificates

As discussed earlier in Chapter 1, “Project overview” on page 3, a Public Key Infrastructure (PKI) is made up of many pieces, products and processes. This section is a synopsis of the process for certificate management:

1. Submit a request for a certificate
2. Submit the request to a Registration Agency (RA) or a Certificate Authority (CA)
3. Obtain the certificate signed by the RA or CA
4. Make sure that the entity at the other end of a connection has a copy of the root certificate of the CA
5. Configure SSL CipherSpecs

3.2 Submit a request for a certificate

Although this looks like a simple “fill in a form” application, there are several parts to the process.

The very first part is finding a piece of software that will generate the request and a certificate from the request. The IBM security manual for WebSphere MQ suggests using the Windows `makecert` command or the UNIX and Windows `iKeyman` utility. Neither of these is really practical because `makecert` is part of the Microsoft Windows Software Development Kit (SDK) which is hard to find for most MQ administrators and supports a 512 bit key of medium strength security. The `iKeyman` software, likewise, is unavailable to most of the expected readers of this redbook.

There are six solutions to this problem that worked for us.

3.2.1 Using the HTTP server solution (iKeyman)

The first solution is to use the Windows `iKeyman` utility that is shipped with the IBM HTTP Server installed as part of WebSphere Application Server or with the IBM Directory (like the Tivoli Access Manager family, Tivoli Identity Manager and Tivoli Privacy Manager). It is also available free of charge and can be downloaded from the IBM Web site:

<http://www14.software.ibm.com/webapp/download>

Once installed, you must click **Start -> Programs -> IBM HTTP Server -> Start Key Management Utility**.

This is the same code base as the UNIX iKeyman code and has the same look and feel.

Most of the cryptography books talk about the password *entropy*. All this really means is that short passwords are easier to guess than long ones, and long ones containing special characters and mixed cases are even harder to guess. iKeyman stores private keys in a database and the password that you associate with this key database. The iKeyman GUI illuminates keys to show you the password strength and we strongly recommend that you use this tool as a guide. Refer to Appendix C on page 389 for further details and illustrations.

3.2.2 Using a CA in test mode

The second solution is to use a CA. Almost all of the certificate authorities allow you to create a certificate request and offer to optionally supply you with a one year certificate as well. Some CAs require more information than others, some require credit card numbers, but they all essentially offer a similar service. We have found the DST CA to be one of the easiest to work with, although this is by no means endorsing one CA over another. Refer to Appendix C on page 389 for illustrations.

3.2.3 Using Microsoft Windows 2000 certificate services

The third solution is to create a CA using Microsoft Windows 2000 certificate services. This is a base component of Windows 2000 Server. Certificates can be requested using a Web browser by loading a request from the key pair, or by completing a form and generating a certificate with an exportable private key. Refer to Appendix C on page 389 for examples.

3.2.4 Using OpenSSL

The fourth solution is to use OpenSSL. Download the toolkit from www.openssl.org. Compile with GNU C compiler and use the command line to generate certificates, or create a wrapper application.

3.2.5 Using Digital Certificate Manager (DCM)

DCM is a core component of OS/400. It supports the cryptographic co-processor and can be accessed using http or https from a Web browser.

3.2.6 Using RACF

The glossary in this redbook explains that some products in the IBM portfolio are being rebranded. The IBM z/OS Security Server is one such product. It was

formerly known as Resource Access Control Facility (RACF) and we have elected to continue to call the product RACF in this redbook to prevent confusion.

The RACF panels can be used to create certificate requests. More importantly, RACF can also be used to generate real certificates, both self-signed and self-administered. Having RACF function as the certificate authority in an enterprise removes the need to negotiate expensive license fees with well known certificate authorities.

The RACF panels and commands we used when implementing SSL for our business scenario are shown in Chapter 8, “Business scenario architecture” on page 141.

3.3 Cryptographic co-processors

The cryptographic co-processor card used in our redbook development is an IBM 4758 PCI card installed in our z/OS environment. All G5 and later processors come delivered with two cards as standard.

This card is essentially a 486 class processor inside a tamper responding environment. This means that if you try to open the card, it will self destruct.

The card is accessed through the PKCS#11 standard API which supports DES, triple DES (with triple-length keys), RSA and DSA, and SHA-1, MD2, and MD5 hashing services. See 3.4, “Algorithms” on page 35 for more details. You can add additional co-processors to a system to further offload cryptographic computations.

As far as WebSphere MQ is concerned, the z/OS implementation requires a number of parameters that are passed to the user through the RACDCERT interfaces. On other platforms, the use of the cryptographic processor is transparent to WebSphere MQ. This is because the WebSphere MQ code layers on top of the GSKit toolkit, and it is the toolkit that interfaces to the card through vendor-supplied PKCS#11 implementations.

Figure 3-1 on page 35 shows what the co-processor card looks like. It is a PCI board that fits in a standard PC style PCI slot. At the time of writing this redbook, these cards were relatively inexpensive.



Figure 3-1 Co-processor card

Offloading processor tasks to this board in a z/OS environment requires microcode configuration and the configuration and starting of the ICSF subsystem. Information about the ICSF can be obtained from:

<http://www-1.ibm.com/servers/eserver/zseries/ebusiness/techinfo/icsf.html>

The cryptographic co-processor is where we recommend stashing private keys when RACF is used as a CA.

3.4 Algorithms

This section describes the algorithms that can be selected when deploying SSL.

3.4.1 CipherSuites and CipherSpecs

A CipherSuite is a set of cryptographic algorithms. A suite contains three distinct algorithms :

- ▶ The key exchange and authentication algorithm
- ▶ The encryption algorithm used to create cipher text from plain text
- ▶ The Message Authentication Code (MAC) used to generate a message digest, which is also known as a *digital hash*

There are various choices for the key exchange and authentication algorithms: the RSA implementation of the Diffie-Hellman exchange, some extremely complicated mathematics called Elliptical Curve Cryptography (ECC), and so on. The list below, for example, addresses the CipherSuites in the Netscape SSL V3 specification.

- ▶ SSL_RSA_WITH_DES_CBC_SHA
- ▶ SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
- ▶ SSL_DHE_DSS_WITH_DES_CBC_SHA

- ▶ SSL_DH_anon_WITH_RC4_128_MD5
- ▶ SSL_FORTEZZA_KEA_WITH_NULL_SHA

The words between SSL_ and _WITH specify the key exchange and authentication algorithms. All the variants of these algorithms are included in SSL V3. The GSKit that WebSphere MQ SSL layers upon only supports the RSA exchange so that is what WebSphere MQ supports. WebSphere MQ supports SSL V3 and ECC is not included in the specification for SSL V3.

A CipherSpec is one step down from a CipherSuite (a subset, if you will) and it identifies the combination of the encryption algorithm and the MAC. Since WebSphere MQ only supports RSA, you will only see the term CipherSpec in the product.

3.4.2 HASH and MAC

Think of a checksum. A piece of data is run through an algorithm and a string of sixteen or twenty bytes is produced. This is called a Message Authentication Code (MAC, also called a *hash*). The idea is that no two messages will create the same MAC. If they do, it is called a *collision* but collisions are very hard to create. This means that, provided you can protect the MAC in flight (which we do with encryption), then the receiver of a message can run the same algorithm, get the string of 16 or 20 bytes and compare the two. If they match, the message has not been altered in flight. Change just a single bit in the message and the sending MAC will not match the receiving MAC.

MD5

The MD5 algorithm was developed by Ron Rivest (the R in RSA) in 1992. It was based on MD4 written in 1990, also by Ron Rivest. The algorithm takes a message of any size and produces a 128 bit (16 bytes) message digest.

SHA-1

The SHA-1 algorithm was developed by the USA National Institute of Standards and Technology (NIST, with a little help from the NSA (National Security Administration)). The initial version was published in 1993 (SHA) and revised in 1995 (SHA-1). This algorithm produces a 160 bit (20 bytes) message digest.

If you look closely at the CipherSpec table in the WebSphere MQ Security manual, you will see that only MD5 and SHA-1 are currently supported for computing MACs.

3.4.3 Symmetric key encryption algorithms

These are the algorithms used to encrypt data that flows over a WebSphere MQ channel once the initial SSL handshake is complete. It is important to note that the SSL handshake only occurs when a channel starts or restarts after being stopped.

Once started, the two ends of a channel will use the same symmetrical key created at channel startup to encrypt every buffer that flows between the message channel agents. Given that the preamble for these packets is in a well known form, it is conceivable that a crypto analysis attack could be mounted that would disclose the symmetrical key given a suitable amount of data (never ending channels for example). Customers may therefore elect to specify a disconnect interval for channels which will force a new SSL handshake, resulting in a new symmetric key being generated.

Symmetric keys come in different sizes, bigger being better for security and worse for performance. The WebSphere MQ supported ones are DES, T-DES (Triple DES), RC2, RC4 and AES.

DES and Triple DES

Although DES was just about secure in the last century, it uses keys that are susceptible to “brute force” attacks because they are only 56 bits long. Triple DES (also a standard as ANSI X5.92) uses three DES operations to provide significantly more security. It uses two keys and performs an encryption with the first key, a decryption with the second key and then a further encryption with the first key.

Since Triple DES uses two keys, the effective key length is 112 bits. This is much more secure than 56 bits since security doubles with each bit you add to the key. If you could break DES by brute force in one second (which is very far from being the case) then it would take 2.285 billion years to break Triple DES, give or take a fortnight.

Some Triple DES operations can be handled in hardware in the cryptographic co-processor on z/OS, making it an attractive choice for certain customers. However, UNIX GSKit does not make use of the data encryption facilities on a cryptographic card even if they are present; it only makes use of hardware assistance on key exchange and authentication algorithms.

RC2

RC2 is a variable key-size block cipher designed by Ronald Rivest for RSA Data Security (now RSA Security). “RC” stands for “Ron's Code” or “Rivest's Cipher”. It can be made more secure or less secure than DES against exhaustive key search by using appropriate key sizes.

RC4

RC4 is a stream cipher (it operates on data one byte at a time). It is used inside Netscape and Internet Explorer for SSL.

AES

AES is the official US government's replacement for DES. WebSphere MQ only supports AES on the AS/400, AIX, HP-UX, and Linux for Intel platforms. Therefore, we have elected not to use CipherSpecs that include AES.

3.5 Lightweight directory access protocol (LDAP)

LDAP is a set of well defined APIs that access a datastore that can be implemented using any technology a vendor wishes to use. LDAP is therefore a directory access mechanism and a directory datastore.

3.5.1 Differences between directories and databases

A directory is often described as a database, but it is a specialized database that has characteristics that set it apart from general purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Hundreds of people might look up an individual's phone number, or thousands of print clients might look up the characteristics of a particular printer. But the phone number or printer characteristics rarely change.

Because directories are meant to store relatively static information and are optimized for that purpose, they are not appropriate for storing information that changes rapidly. For example, the number of jobs currently in a print queue probably should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer could contain the network address of a print server. The print server could be queried to learn the current queue length, if desired. The information in the directory (the print server address) is static, whereas the number of jobs in the print queue is dynamic.

As far as WebSphere MQ security is concerned, LDAP is only used to store revoked certificates in a structure known as a Certificate Revocation List (CRL).

You can secure the CRL and the LDAP server to prevent the security risks listed below.

- ▶ The risk of someone removing revoked certificates without authority (
- ▶ The risk of someone revoking certificates without authority (a form of Denial Of Service attack)
- ▶ The risk of someone re-directing the LDAP calls to another LDAP server, either by intent or by accident, potentially leading to the risk of not communicating with, for example, Honest Bob's CRL and instead being directed to Untrustworthy Tom's CRL.

3.6 KEYS and passwords

In many of the security forums, you see questions about key strength. Following is a very short section to remind the reader about binary arithmetic.

Remember that 0001 is binary for one; when we add one, we get 0010 (decimal 2), then 0011 (decimal 3) then 0100 (decimal 4) and on up to 1000, which is decimal 8. So now, remember that 10000 (five bits) is decimal 16, 100000 (six bits) is decimal 32 and so on. So, with the addition of every bit, we double the possible combinations of ones and zeros that make up a numeric representation.

That's why a key length of 57 bits is twice as strong as a key length of 56 bits; there are double the number of possible combinations.

3.6.1 Password strength

Passwords are also just strings of bits. If you have a one byte password, then there are only 256 possible combinations of ones and zeros that makes up your password (8 bits in a byte). A brute force attack at guessing this password would not take much time to complete. You start guessing by building the binary string 00000001, then try 00000010, then 00000011 and long before you get to 11111111, you will have guessed your password.

Cryptoanalysis just means using intelligence to speed up the process of finding out keys and passwords. If you assume that you can only type the keys you see on the keyboard (not a bad assumption) then you can rule out certain bit combinations, for example, 00000000 and 11111111, thus making your attack quicker to accomplish.

The point of all of this is that if you protect keys in a key ring or key repository and protect them with a password, then it does not matter how big the keys are; they could be a million bits long. What matters is the length of the key to the key ring, which in our case is a password. Imagine having a key to Fort Knox and then

putting it in a piece of luggage secured with a tiny suitcase lock. The big key is at risk.

Passwords that you choose to secure your key rings and key stores are at least as important as the length of the keys that you use in cryptography. Also, the length of a password is not the same as the length of a key, because the artifact creating a key will use all possible bits and you will not. This is what is meant when it is said that there is a lot of entropy in passwords.

So, if you do everything in 8 bit chunks, then a 160 bit key fits in 20 bytes and will require a password bigger than that to get around the entropy problem, so because there are very few words that are bigger than twenty bytes), we need to use a pass phrase which is, of course, just a collection of words.

Keep this in mind when we start securing key stores later in this book.

3.7 SSL setup for WebSphere MQ

To secure our WebSphere MQ environment further and implement the SSL function, the following tasks were defined and performed in the procedures listed below:

- ▶ Planning for SSL
- ▶ Preparing certificates
- ▶ Websphere MQ objects for SSL

3.8 Planning for SSL

Before going into composition, these items are determined indispensable.

- ▶ Type of CipherSpecs (encryption and hash function)
- ▶ The key repository file name which will store the certificates
- ▶ Preparing for the required certificates

These are additional options that can be considered in the setup:

- ▶ SSL client authentication
- ▶ DN filtering
- ▶ CRL check by LDAP
- ▶ Using cryptographic hardware

3.9 Preparing certificates

Each queue manager and WebSphere MQ client has its own digital certificate. On all platforms, this certificate is stored in a key repository using the digital certificate management tool, for example, z/OS (RACF), UNIX (iKeyman) and OS/400(DCM).

3.9.1 Key repository

You have to prepare the key repository for storing certificates. It is referred to differently on these named platforms:

- ▶ Keyrings in RACF, ACF or TopSecret on z/OS
 - ▶ Key database on UNIX and OS/400
 - ▶ Key store or certificate stores on Windows
- Private keys are stored in the Windows registry.

3.9.2 Defining the certificate type for your system

There are two types of certificates, self-signed and CA-signed.

Self-signed certificate

If you use self-signed certificates, you must store the certificates for your systems and all the other systems they connect to as illustrated in the figure below.

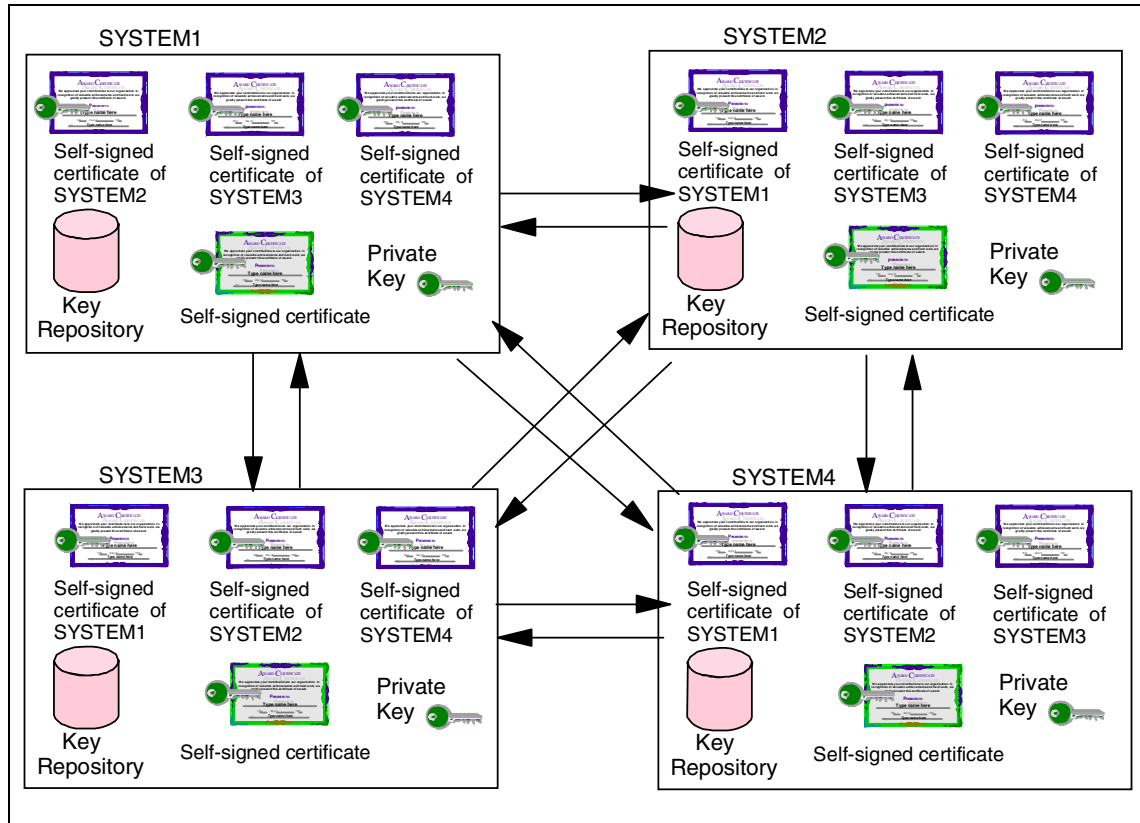


Figure 3-2 Management with self-signed certificate

CA-signed certificate

To use a CA-signed certificate, you need to send a certificate request to a CA. Then get back the CA's certificate and your personal certificate signed by the CA. From the system management point of view, it is advisable to choose a CA-signed certificate because it saves you from having to distribute the certificates for your systems to all the other systems they connect to.

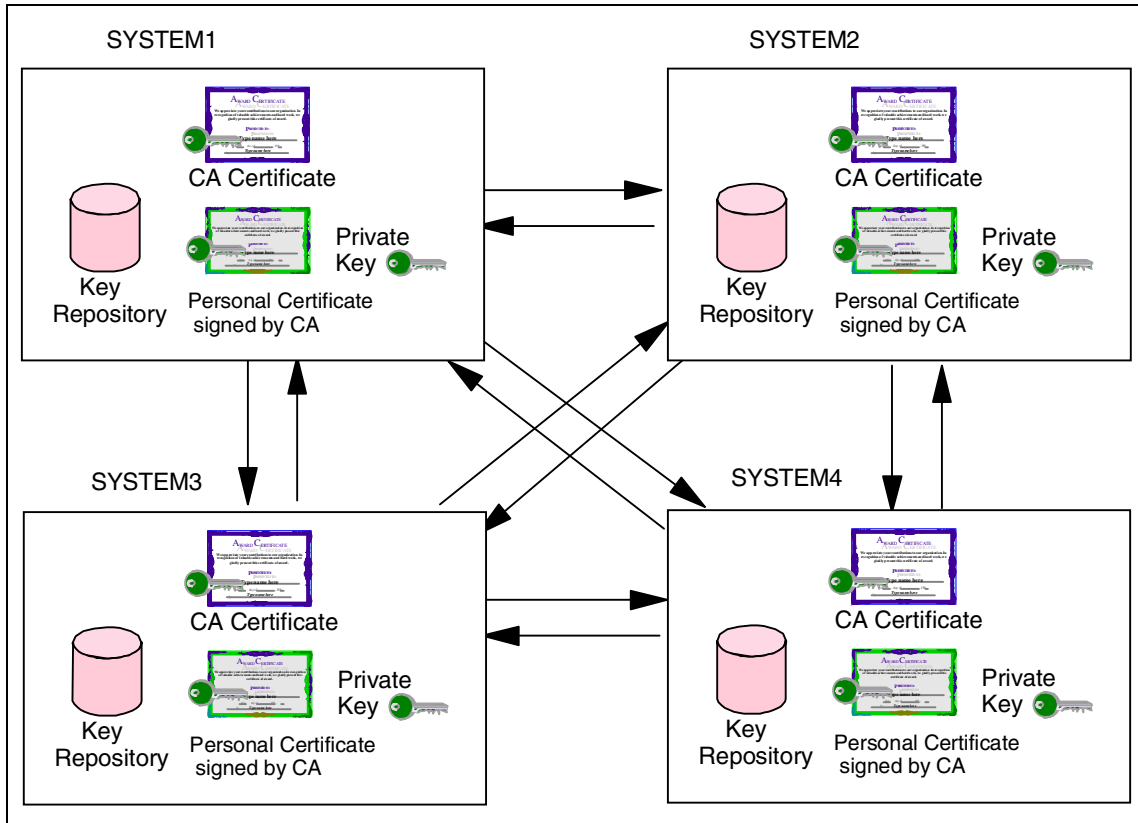


Figure 3-3 Management with CA-signed certificate



Platform security

This chapter discusses what technologies and procedures are available today to assist in securing an enterprise. The four most commonly used platforms, z/OS (zSeries), OS/400 (iSeries), IBM AIX (pSeries) and Windows 2000 (xSeries), are examined here.

4.1 z/OS security

This section discusses z/OS security.

4.1.1 Overview of z/OS security

There are many elements required to make up a successful security solution; among them are the following.

- ▶ Strong hardware isolation functions and system integrity to ensure that misbehaving or malicious applications and users cannot affect other applications or users.
- ▶ System-level security to control and monitor the actions of users and applications on the system.
- ▶ Network-level security to protect your system from outside attackers on the Internet.
- ▶ Transaction-level security to provide protection for business transactions on the Internet.
- ▶ Integrated cryptography support which underlies a solid network and transaction-level security implementation, and aids in providing system-level security.
- ▶ Integration of the system security functions with system-provided applications, vendor-provided applications, and your own applications.

The combination of z/OS hardware and z/OS software along with good working security processes can provide a way of protecting systems.

For further details on all of these areas, see:

<http://www-1.ibm.com/server/eserver/zseries/zos/security/>

The following is a brief summary highlighting some of the areas covered in more detail on this Web page.

- ▶ Basic isolation functions
 - The basic isolation functions keep users separated from each other and from the operating system running on the hardware. The functions include:
 - Storage protection keys
 - Authorized versus unauthorized execution states
 - Multiple address spaces
 - z/OS takes full advantage of these functions to help ensure the integrity of the system and your data.

- ▶ System-level security gives you the ability to identify and authenticate users of the system, control the access they have to system resources, for example databases, transactions, and programs, and audit the use of those resources. Ways of doing this include:
 - User ID and password, or user ID and PassTicket authentication
 - OSF Distributed Computing Environment (DCE) credentials or Kerberos Version 5 credentials
 - Industry-standard X.509 digital certificates with applications such as DominoGo Webserver™
- ▶ Network level security

To protect a machine from network attacks, separate programs or machines known as *firewalls* are often used; they use a variety of technologies to isolate the protected machine from the attackers, while still allowing everyday network traffic. These firewall technologies include:

 - Packet filtering to limit the kind of network requests that can reach your machine
 - Proxy and SOCKS servers to control TCP/IP connectivity
 - Domain name services to hide information about your private network from users or programs on the public network
 - Encryption of TCP/IP traffic (TCP/IP tunnels, or Virtual Private Networks (VPN)) to allow private communication over the public network

The various components of z/OS can be used together to provide the different types of protection that would otherwise require a separate firewall machine or program.

- ▶ Transaction-level security allows two entities on the Internet to conduct a transaction privately and with authentication.

Two leading technologies provide transaction-level security today:

 - Secure Sockets Layer (SSL), a de facto standard developed by Netscape communications corporation and generally used by Web browsers and servers, provides a private channel between the client and server that ensures privacy of the data, authentication of the session partners, and data integrity for the messages. SSL involves the use of both public- and private-key-based encryption.
 - Secure Electronic Transaction™ (SET™) is an open standard, multi-party protocol used for conducting secure bank card payments over the Internet. SET also provides authentication, data integrity, and data privacy, again through the use of encryption technology.

Integrated cryptography provides the fundamental encryption and decryption operations required for SSL and SET requests and allows implementation of the VPN connections provided by z/OS firewall technologies for network-level security.

4.1.2 z/OS Security Server

The z/OS Security Server provides the following:

- ▶ RACF
- ▶ DCE security server
- ▶ Firewall technologies
- ▶ Lightweight directory access protocol (LDAP)
- ▶ Open cryptographic enhanced plug-ins (OCEP)
- ▶ Network I authentication service

which includes supports for X509 certificates and PKI, Kerberos, DCE, user ID/password, user ID/PassTicket, SSL, and LDAP for authentication.

The Security Server also provides facilities for applications to perform access checking, auditing, and encryption.

4.1.3 Resource access control facility (RACF)

The primary component of the Secureway Security Server for z/OS is the Resource Access Control Facility (RACF). It controls what you can do on the z/OS operating system and can be used to protect your resources. It does this by providing the following mechanisms.

- ▶ Identification and verification of users

A user is first defined to RACF by the installation's security administrator who assigns the user a unique user ID and a temporary password. The temporary password enables the user to log on to the system for the first time. As soon as the user logs on for the first time, RACF requires the user to supply a new password of their own choice. The administrator may set the password to expire after a certain time interval; if this is the case, it will have to be changed periodically.

When you log on to the operating system, RACF requires that you provide a user ID (a unique identifier). RACF then verifies that you are the user you say you are by asking for a password it can check against the user ID. Each RACF user ID has a unique password. This password should be protected by you by keeping it secret. As with any user ID and password combination, if it is compromised, it will no longer be considered secure.

- ▶ Authorizing users to access protected resources

Installations can also use RACF to define what authority each user ID has, or what authorities a group of user IDs have. A group is a set of individuals who have common requirements. For example, a group of testers may all need the same access to certain datasets.

With this, RACF controls what each user can do on the system. Some individuals have a great degree of authority, while others have little authority. The level of authority a user is given is based on what they require in order to do their job.

RACF also provides a mechanism that allows you to protect resources. A resource is any information that is stored in an installation's computer system, for example, data sets.

RACF stores all the information it has about users, groups, and resources in profiles. A profile is a record of RACF information that has been defined by the security administrator. There are user, group, and resource profiles.

Whether a user has authorization to access certain resources is determined by RACF using the information it has stored in the profiles. User profiles provide user attributes. User attributes describe what system-wide and group-wide access privileges a given user has to protected resources.

A group profile describes the kind of authority a user has (as a group member) to access resources that belong to that user's group.

Resources themselves have profiles describing the type of authority needed to use them.

The security administrator or someone in authority in your organization controls the information in your user profile, in group profiles, and in resource profiles. You, as an end user, control the information in profiles describing your own resources, such as your own data sets. You can protect your data by setting up resource profiles.

A resource profile can contain an access list as well as a default level of access authority for the resources it protects. The access authority that specifically names users and groups is held in the access list. The default level of access authority associated with the resource for that profile applies to anyone else who is not on the access list.

- ▶ DL Auditing access attempts

RACF provides a way to record what is done on the system. It keeps track of what happens on the system so that an organization can monitor who is logged on the system at any time. RACF reports whether anyone has attempted to perform unauthorized actions. For example, RACF can record that someone has tried to access or change data to which they do not have the correct authority.

4.2 OS/400 security

This section is a summary of OS/400 security for readers who are not familiar with its features.

The iSeries and its operating system OS/400 have evolved from the system/36 and the system/38, and have an enhanced combination of both of their original security features.

On OS/400, users are defined with profiles and can be organized into groups. These can be given special capabilities and limitations. The OS/400 has a number of profiles provided as part of the operating system, only QSECOFR (security officer) is intended for sign on.

User profiles contain a list of objects the user owns or is authorized to use. For objects owned by a user, the profile also lists the other users' authorizations to those objects.

Group profiles are used to simplify the administration of object authorities by authorizing users through a smaller number of group entities. OS/400 does not treat groups any differently than user profiles, so it is necessary to set the group profile password to *NONE to prevent sign-on.

There are three distinct types of security on OS/400 relevant to WebSphere MQ.

1. Object security
2. System security
3. Physical security

4.2.1 Object security

The term *object* is used to refer to a number of different kinds of information on the system that can be accessed via the standard OS/400 interface.

Every object on the system has an owner, who has an important role in the security function. The object owner and security officer can grant or revoke authority to an object to other users.

It is important to note that there is a difference between authority to an object and authority to the data contained within. Operations such as moving, renaming, saving, or deleting apply to the object and it is possible to have authority for these operations without having access to the data stored in the object. Similarly, one can have full access (read, write, update, delete, execute) to the data in an object without having full authority to manipulate the whole object.

Certain programs or commands called by a user may require a higher level of authority than is normally available to that user. Adopted authority allows a user to temporarily gain the authority of the owner of a program, in addition to the user's own authorities, while that program is running. This provides a method to give a user additional access to objects, without requiring direct authority to objects.

4.2.2 System security

System security is an integrated function of the OS/400. Users are identified and authenticated by a single security mechanism, at the system level, for all objects, functions and environments.

Storage allocation on the OS/400 is different from most other computer systems'. The OS/400 uses a shared storage system in which all portions of main and auxiliary storage are addressed as though they are within a single area (or level). The system uses the object name to determine where it exists in storage. This means that the user can find objects by name, rather than by storage locations. Because operations cannot be performed on an object that is not in main storage, the system moves all or part of the object into main storage as it is needed, and moves it back into auxiliary storage when the object is not needed.

The OS/400 has over one hundred variables that control system-wide functions. These are called system values. Some of the system values are security-related.

The security-related system values fall into four main categories:

1. General security defaults
2. Audit control
3. Password rules
4. Other system values related to security

4.2.3 Physical security

Physical and procedural security controls provide the basis upon which other controls (such as software security) are built. In addition to physical access control and output distribution procedures, the iSeries has two unique hardware features, which are important for physical security:

- ▶ System keylock - to enable or disable certain system service functions
- ▶ Display station functions - keylock and play/record keys

The IBM common cryptographic architecture services/400 PRPQ is used for hardware cryptography on the iSeries. The hardware encryption feature provides the following:

- ▶ Data Encryption Standard (DES)
- ▶ Public Key Algorithm (PKA)
- ▶ Key management functions
- ▶ Security Application Programming Interfaces (SAPIs) to invoke cryptography
- ▶ Functions from user-written applications
- ▶ Security for the IBM 4700 banking terminal

4.2.4 Further reference

- ▶ *An Implementation Guide for AS/400 Security and Auditing: Including C2, Cryptography, Communications, and PC Connectivity*, GG24-4200
- ▶ *AS/400 Tips and Tools for Securing Your AS/400*, SC41-5300-04
- ▶ *OS/400 Security - Reference V4R5*, SC41-5302-04
- ▶ *AS/400 Internet Security: Protecting Your AS/400 from HARM in the Internet*, SG24-4929

4.3 AIX security

UNIX security is based on three main services:

1. User management and authorization
2. Network security
3. Trusted Computing Base (TCB)

4.3.1 User management and authorization

AIX is a multi-user system, like other UNIX systems. It is managed by user IDs and group IDs. Before a user logs on to a UNIX system, the administrator (called root) has to create a user ID for this user and set an initial password. The password must be changed at the first login. A user must belong to one or more groups.

In UNIX, all user ID information, including passwords, is stored in a text file named `/etc/passwd` or NIS (Network Information System). In UNIX `/etc/passwd` is also encrypted in the `/etc/security/passwd` file and the `/etc/security` directory is

limited to access. 5L now supports LDAP and the Kerberos 5 module, providing more robust and thorough user management.

Authentication

A user is authenticated by a user ID and a password in order to use an AIX system. When you log on to an AIX system, you are first prompted for a user ID, then for a password. AIX verifies the ID and password by using the password file, LDAP or Kerberos.

The administrator can set many options relating to user security, such as user expiration date, password expiration, password rules and so on.

Authorization

After you are authenticated, authorization for the resources, files and directories is granted by using your user ID. AIX provides standard UNIX file permissions, read (r), write (w) and execute (x), for owner, group and others.

In addition, AIX also uses the Access Control Lists (ACL) to further define access permissions to files and directories. It provides more strict authorization rules for individual users. However, by default, AIX does not use ACL.

Considerations for WebSphere MQ

Before you install WebSphere MQ for AIX, you must create the mqm user who can belong to the mqm group on your local machine or NIS server machine. All files, libraries and directories for the product are owned by mqm user. The following directories are created automatically in your installation.

- ▶ /usr/mqm : product code, manuals
- ▶ /var/mqm : working data (queue manager's log and configuration data)

The user who wants to administer WebSphere MQ queue manager must belong to the mqm group. Since mqm has the highest authority to manage the objects, you should be very careful with the mqm user's password.

4.3.2 Network security

When you connect to the Internet, the host is more vulnerable to an attack from a hacker. Hence, network security becomes more important than ever. The security policy for the network is an extension of the operating system's components, such as:

- ▶ Connection authentication
- ▶ User authentication
- ▶ Access control for files and directories (data import and export)

Transmission control protocol/Internet protocol (TCP/IP)

AIX supports TCP/IP V4. TCP/IP V6, the next generation, will be fully supported by AIX.

In AIX, general TCP/IP security rules apply. The **securetcpip** command disables nontrusted daemons and applications.

AIX supports all the remote commands and TCP/IP applications. For a virtual terminal, **telnet** and **rlogin** are provided. These commands use a password for user authentication and since the password flows in clear text over the network, it is a security weakness. Therefore, using SSL is recommended to overcome this exposure.

TCP/IP security

AIX provides the following TCP/IP security:

- ▶ On-demand tunneling

The functions for negotiating, computing and refreshing session keys will only be performed when necessary.

- ▶ TCP/IP address filtering
You can configure filters using a Web-based system manager from a remote machine.
- ▶ Certificate-based use of digital signature for Internet Key Exchange (IKE)
Authentication is accomplished by signing IKE messages using X.509 certificates.
- ▶ SOCKS API
By using AIX SOCKS API, the generic TCP/IP applications can connect to hosts through a TCP/IP proxy using SOCKS protocol V5.

To help you pinpoint configuration failures and determine security attacks, the audit logs for TCP/IP security have been improved on AIX 5L™.

Network file system (NFS)

AIX supports NFS V3, the latest version of NFS and NFS V2 for backward compatibility with existing NFS clients. Standard NFS security rules are written in the `/etc/exports` file. It contains file names exported to be mounted from remote systems, permissions (read-only or read/write), and privilege for root access.

In addition, AIX provides ACLs between AIX systems. It offers this using secure RPC for integrity and authentication, but since it contains Data Encryption Standard (DES), it is restricted outside the U.S.

Distributed computing environment (DCE)

The DCE is a standard solution for creating a client/server application environment. AIX supports DCE security.

4.3.3 Trusted computing base (TCB)

TCB can limit access of a system resource to users, applications and processes. It is a useful tool to detect system resource changes, such as files, programs and devices(/dev). The system administrator can set particular programs and shell scripts as trusted by TCB.

You can only install TCB during your initial AIX installation on your system. You should decide whether or not to use TCB before your AIX installation, since TCB is not installed by default and you cannot add it without an AIX reinstallation.

TCB monitors system, user and devices files based on its configuration file, `/etc/security/sysck.cfg`.

The system administrator can set additional restrict access control information to the file using the **tsbck** command. This **tsbck** command has three basic operations: add, remove and check.

For example, you can add your program to TCB and no one can make changes without your knowledge.

It is important to plan how frequently you check your modified files and how many files should be monitored in your system, because the system has an important impact on performance. We recommend that all the AIX files and any programs that the root user is going to run be monitored. You will be notified if someone updates your system configuration.

Trusted communication path

- ▶ The tsh

The tsh shell allows you to run programs which have the TCB mode set. If someone creates a malicious program, it will not run because it does not have the TCB mode. You can detect if the files are changed using the **tcchk** command.

- ▶ Secure attention key (SAK)

SAK is a reserved key sequence and restricted to login from the trusted devices. Note that SAK only works with locally attached devices.

4.3.4 Further reference

- ▶ *AIX 5L and Windows 2000: Side by Side* (SG24-4784-02)
- ▶ *AIX 4.3 Elements of Security Effective and Efficient Implementation* (SG24-5962-00)
- ▶ *Additional AIX Security Tools on IBM eServer™ pSeries, IBM RS/6000, and SP™/Cluster* (SG24-5971-00)

4.4 Windows 2000 security

There have been a number of key improvements in Windows 2000 over Windows NT 4.0. Windows NT in the enterprise relied on a model of domains. Users and groups were given access to resources throughout the domain. The users, groups and passwords were held on domain controllers. The fundamental weakness of the security in Windows NT was the NT Lan Manager (NTLM) protocol that was used to authenticate users. Added to this was an immature TCP/IP protocol stack that was not ready for the onslaught of the Internet. Windows 2000 is considerably better. If you are running WebSphere MQ on a

Windows platform in the enterprise and you are concerned about security, you should be running Windows 2000 as a minimum.

Windows 2000 security consists of:

- ▶ Users and groups
- ▶ Access control to resources
- ▶ Network authentication using a directory service and Kerberos
- ▶ An encrypted file system
- ▶ Support for PKI, including the ability to create a certificate authority (CA)
- ▶ Auditing function
- ▶ Centralized security policy management

4.4.1 Local logon process

Security on Windows 2000 relies on authenticating a user by user name and password. On a Windows 2000 workstation there is a local security database held in the registry. The local security database contains local users and groups. The local security database is accessed using the Security Accounts Manager (SAM). The logon process is as follows:

1. The user presses **Ctrl+Alt+Delete** to start the logon process. This is to prevent the logon process from being replaced by a spoof program such as a trojan.
2. The user types in the user name and password. This information is collected by the module known as Graphical Identification and Authentication (GINA).
3. The GINA then passes the information to the next security layer, which is the Local Security Authority (LSA), for checking.
4. The LSA in turn passes the information to the Security Support Provider Interface (SSPI).
5. If this was a logon to a domain, the SSPI would communicate with a Kerberos server. Since this is a local logon and no Kerberos server is available, there are some internal fallbacks that occur that actually transpire into the user information being passed back up the stack to the GINA and back down to the SSPI. However, the end result is that SSPI communicates with the NTLM driver that is a legacy from Windows NT.
6. The NTLM passes the user name and password to the Netlogon service.
7. The Netlogon service authenticates the user with the local security database by passing the user information to the SAM.
8. If the logon information is incorrect, an error message is passed back.

Security identifiers

Each principle in Windows 2000 is represented by a unique identifier known as a Security Identifier (SID). Users, groups, computers, domains and so forth each have an associated SID. When you create a user, you create an associated SID. If the user name is renamed or the password changes, the SID persists.

However, if the user name is deleted and recreated with the same name and password, the SID will be different. If you create identically named accounts on two machines, for instance machine A and machine B, the user accounts will not have the same SID. For example, if a user called ALICE on machine A is created with password *wonderland* it will not have the same SID on machine B.

This is important in the context of WebSphere MQ since the SID is passed in the MQ Message Descriptor (MQMD).

When a user logs on to Windows 2000, an access token is created. The access token is a combination of the user account SID, the SID of the groups that the user belongs to, and a Locally Unique Identifier (LUID). The access token is the token that is checked when accessing resources.

The weakness of Windows 2000 in a workgroup is the compatibility with NTLM. For a secure environment, WebSphere MQ should run on a Windows 2000 server with a domain held in a directory.

4.4.2 Active directory

The active directory represents the major change from Windows NT. The active directory organizes resources in a directory structure derived from the X.500 model. The active directory allows users, groups and resources to be managed from a central location, grant or deny access rights, and delegate security administration. It is part of the Windows 2000 TCB.

Objects in the active directory are stored in a hierarchical, object-oriented manner. The active directory provides multi-master replication to support distributed network environments.

Containers are used to represent a collection of related objects. Given that a domain is a single security boundary of a computer network, the active directory could be said to be made up of one or more domains.

On a standalone workstation, the domain is the computer itself, whereas in a network, a domain can span several physical locations.

One or more directory partitions make up the active directory. Directory partitions are contiguous subtrees of the directory that form a unit of replication. This means that any given replica is always a replica of some directory partition.

The Global Catalog (GC) holds a replica of every object in the active directory but only a scaled down version with a small number of each object's attributes (those most often searched for). This allows users and applications to find objects in an active directory domain tree without knowing what domain it belongs to. This Global Catalog is built automatically by the active directory replication system.

One of the most important security features of the active directory is *delegation*. Rather than having domain administrators with complete authority over large segments of the user population, delegation allows a higher administrative authority to grant specific administration rights to highly-trusted individuals or groups. Windows 2000 defines many specific permissions and user rights for this purpose. Using a combination of group membership and permissions, the most appropriate role for a person can be defined.

Examples of specific permissions that might be delegated by the administrator are resetting users passwords or creating a new user.

With a number of different authentication methods, active directory provides an increased level of security for Windows 2000 systems. Once a user is logged on, all of the system resources are protected through a single authorization model.

Further information on the active directory can be found in the Microsoft documentation at:

<http://www.microsoft.com/windows2000/techinfo/howitworks/default.asp#section2>

4.4.3 Network authentication

Windows 2000 actually supports a number of logon protocols which can be set by policies, such as using PKI certificates. However, the default protocol for a network logon is to use the Kerberos authentication protocol. This replaces the NTLM used in previous versions of Windows. Windows supports Kerberos V5 authentication. This is defined in RFC 1510, available at:

<http://www.ietf.org/rfc/rfc1510.txt>

When a user logs on to a Windows 2000 domain, an active directory server is located. The Kerberos authentication service is used to authenticate the user against the active directory and provide access control to resources.

Regardless of the process used to authenticate the user, the end result is a set of credentials that are used throughout the domain. The credentials are session-based and refreshed at each session. The credentials, which consist of a session key as well as the SID, are used to gain access to resources.

The usage of Kerberos in Windows 2000 is explained in the white paper *Windows 2000 Kerberos Authentication* at:

<http://www.microsoft.com/windows2000/docs/kerberos.doc>

4.4.4 Group policies

Group policies can be used to manage security settings for objects in an active directory. For example, you can use group policies to assign logon scripts to users and groups, configure security options such as the logon protocol and control auditing across on machines. Group policies can be associated with all objects in an active directory container. Group policies can have far reaching consequences across the whole domain and are a powerful way of controlling security in the enterprise.

Note: Group policies are explained in full in the Microsoft white paper *Windows 2000 Group Policy* available at:

<http://www.microsoft.com/windows2000/techinfo/howitworks/management/group/polwp.asp>

4.4.5 Access control

Each object in Windows 2000 has an associated Access Control List (ACL) that determines what principles have what rights to that object. An ACL is a list of Access Control Entities (ACE) that allow or deny access rights to individuals or groups. It is stored as a binary value called a *security descriptor*. The security descriptor has four parts: Owner, Group, Discretionary Access Control List (DACL) and the System Access Control List (SACL). Each access control entry in the DACL and the SACL consists of a security identifier and an access mask. The access mask identifies the services of an object that may be accessed by storing the appropriate SID. Access is allowed when the access token presented contains a SID that matches a SID contained in the DACL. The SACL is used when auditing is required, in which case the actions of the SID on the object will be audited.

Once a user has been authenticated either on the domain or locally, Windows 2000 access control is managed by the Local Security Authority (LSA) and the Security Reference Monitor (SRM). The SRM enforces the access control list.

When a request is made to open a resource, the security reference monitor compares the access token of the process with the DACL.

Note: The security architecture of Windows 2000 is explained at:

http://www.microsoft.com/windows2000/techinfo/reskit/en-us/default.asp?url=/windows2000/techinfo/reskit/en-us/distrib/dsbg_dat_doqz.asp

4.4.6 NTFS and the encrypted file system

NTFS is the file system used in Windows 2000 to secure objects stored on non-volatile storage mediums. NTFS allows administrators to control access to objects by assigning an Access Control List to the object. The NTFS system was present in Windows NT, although Windows 2000 has improved upon it.

Windows 2000 has the additional ability to encrypt parts or all of a disk. It will encrypt and decrypt files on the fly, virtually invisible to the user. The Encrypted File System (EFS) can be turned on easily by checking an attribute of a file.

Although the idea behind EFS is a good one, a number of flaws exist. One of the weaknesses is that EFS always makes a key recovery agent (the local administrator). Therefore, the administrator always has access to encrypted files, whether the user wants it or not. Another weakness is that keys are stored in the registry where they can easily be recovered and files decrypted.

Overall, EFS is a good idea that is weakened by design flaws. A security professional is likely to have no problem bypassing the EFS and recovering the plaintext. It is useful in some circumstances if implemented with care.

Note: EFS is discussed at:

<http://www.microsoft.com/windows2000/techinfo/howitworks/security/encrypt.asp>

4.4.7 Public key infrastructure and certificate services

Windows 2000 supports a PKI and usage of certificates. PKI certificates can be used to authenticate external users. PKI provides additional strength to the Kerberos authentication protocol. PKI could be considered even for internal authentication, especially in conjunction with smart cards, where high security levels are required. Examples of where PKI certificates can be used in Windows 2000 are:

- ▶ SSL communications
- ▶ Local and network logon
- ▶ Encrypting files on the hard disk using EFS
- ▶ Secure e-mail.

Note: PKI in Windows 2000 is explained at:

<http://www.microsoft.com/windows2000/techinfo/planning/security/pki.asp>

Windows 2000 has the ability to operate as a Certificate Authority (CA). This will enable you to generate certificates for testing and possibly for production systems without necessarily relying on an external CA. See Appendix C on page 389 for details.

The CA, however, is responsible for identifying and vouching for the identity of certificate holders. A CA is also responsible for revoking certificates, should they become invalid.

Note: A step-by-step guide to setting up a certificate authority on Windows 2000 can be found at:

<http://www.microsoft.com/windows2000/techinfo/planning/security/casetupsteps.asp>

4.4.8 Auditing

Windows 2000 allows the monitoring of security-related events. A security log is generated so that the events can be reported. It is also possible to generate an audit trail to track the security administration events on the system.

Before auditing is implemented, an auditing policy must be decided upon. This is to identify what you want to audit. You can choose to audit the success or failure of the following categories:

- ▶ Audit account logon events
- ▶ Audit account management
- ▶ Audit directory service access
- ▶ Audit logon events
- ▶ Audit object access
- ▶ Audit policy change
- ▶ Audit privilege use
- ▶ Audit process tracking
- ▶ Audit system events

By default, these are all turned off when Windows 2000 is first installed. Auditing a local object will create an entry in the security log. The entries that appear in this log will depend on the auditing categories selected for your auditing policy. Although setting up the auditing policy has changed since Windows NT V4.0, the security log is still viewed with the Event Viewer. The following events can be audited:

- ▶ System restart
- ▶ System shutdown
- ▶ Authentication package loading
- ▶ Registered logon process
- ▶ Audit log cleared
- ▶ Number of audits discarded
- ▶ Logon successful
- ▶ Unknown user name or password

4.5 WebSphere MQ security for z/OS

This section discusses security for z/OS WebSphere MQ.

4.5.1 Overview

WebSphere MQ for z/OS uses the z/OS System Authorization Facility (SAF) to provide access control services within the z/OS environment. This is the standard mechanism of providing security in a z/OS environment and has two significant advantages; firstly, there is a security manager for the entire z/OS environment and secondly, there is a choice of External (to WebSphere MQ) Security Manager (ESM) to choose from, providing greater flexibility.

The main ESMs used by z/OS users are Resource Access Control Facility (RACF), Top Secret and Access Control Facility (ACF2).

Throughout this book, the ESM used by WebSphere MQ for z/OS is RACF.

ACLs are implemented as a set of profiles. A user is granted access to a particular profile to allow access to the protected resource. To contain the profiles, WebSphere MQ has a set of RACF classes for its use. These classes are:

MQADMIN	Contains profiles for administration functions, command resources, context and alternate user profiles
MQCONN	Contains profiles to limit connection to the MQ subsystem
MQCMDS	Contains profiles for command security
MQQUEUE	Controls queue resources
MQPROC	Controls process resources
MQNLIST	Controls namelist resources

These lists are defined to the RACF system and need to be defined to other security manager products as well.

The classes need to be activated before security checks can be made, you can do that using the following RACF command:

```
SETROPTS CLASSACT(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCMDS,MQCONN)
```

You should also allow the classes to accept generic profiles using the following RACF command:

```
SETROPTS GENERIC(MQADMIN,MQQUEUE,MQPROC,MQNLIST,MQCMDS,MQCONN)
```

Figure 4-1 on page 65 shows an overview of WebSphere MQ for z/OS prior to V5.2 where each queue manager stored data on local pagesets only accessible to the owning queue manager. It also shows possible connections to the queue manager by CICS, IMS™, Batch and the channel initiator address space (CHINIT, often referred to as the *MOVER*). The mover is the main mechanism used to connect into WebSphere MQ on z/OS from other WebSphere MQ queue managers on other platforms.

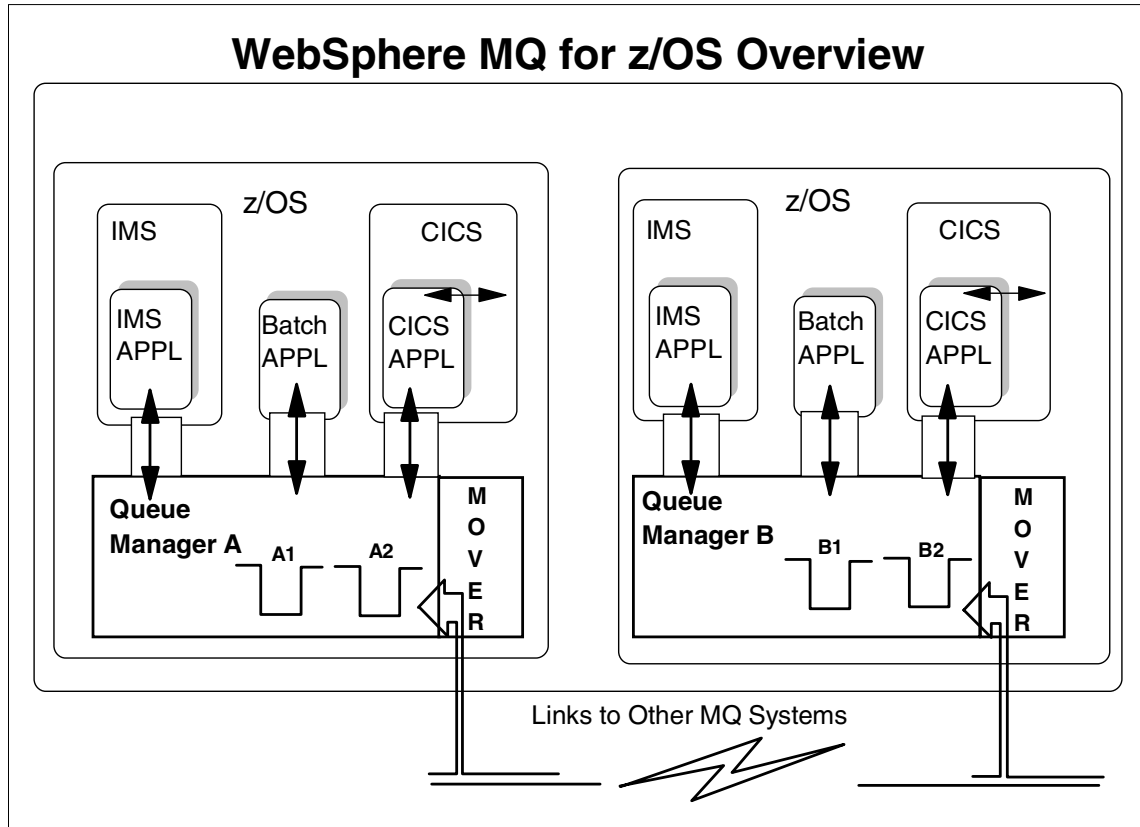


Figure 4-1 Overview of WebSphere MQ for z/OS prior to V5.2

4.5.2 Switch profiles

Switch profiles are RACF profiles which control the level of security checking carried out by WebSphere MQ.

The switch profiles allow a granular control of security checking within WebSphere MQ for z/OS. The switches available prior to V5.2 are:

- ▶ hlq.NO.SUBSYS.SECURITY
- ▶ hlq.NO.CONNECT.CHECKS
- ▶ hlq.NO.QUEUE.CHECKS
- ▶ hlq.NO.PROCESS.CHECKS
- ▶ hlq.NO.NLIST.CHECKS
- ▶ hlq.NO.CONTEXT.CHECKS

- ▶ hlq.NO.ALTERNATE.USER.CHECKS
- ▶ hlq.NO.CMD.CHECKS
- ▶ hlq.NO.CMD.RESC.CHECKS

These are all defined in the MQADMIN class.

If the MQADMIN class is not present or not activated (or there is no external security manager) then security checking is deactivated.

Generic switch profiles such as hlq.NO.** are not detected by WebSphere MQ. The use of switch profiles is extended in WebSphere MQ for z/OS V5.2 to support use of queue sharing groups and shared queues.

Throughout this book, the high level qualifier “hlq” is used as a prefix for profile names defined to RACF. Before WebSphere MQ for z/OS V5.2 this stood for “sub-system id” and was the name of the queue manager to which the profile was relevant. From V5.2 on, the hlq can stand for either “sub-system id” or “queue sharing group id”.

These profiles are not used in the same way that profiles are normally used for access control. They are used simply as switches to activate/deactivate access control checking for various components of queue manager processing. Thus, user IDs are not permitted various access rights to these profiles.

The default action for WebSphere MQ is to have access control checks activated. The presence of a switch profile will deactivate security checking for the appropriate component. Since these switch profiles are not present by default, it means that explicit action is required to deactivate security processing within a WebSphere MQ environment, once RACF is active and the WebSphere MQ classes are defined (no checking is possible otherwise).

If the hlq.NO.SUBSYS.SECURITY profile is present then no further checks are performed for other switch profiles.

Activating security support in this way means that no security system management is performed from within the queue manager, which is deemed to be a good thing. However, the use of profiles in this way is quite unusual.

The different switches represent the different components of WebSphere MQ to which access control checks may be applied and are split into three areas:

1. Connecting to the queue manager
2. MQ API
3. MQ commands

Command security and command resource security checking (the latter part, above) were originally designed to be used together to provide a greater granularity to determine whether the issuer of the command was allowed to issue it, whether that command affected a resource, and whether they were allowed to issue that particular command for the given resource. If both are active, then all commands issued that affect resources will have both security checks carried out and must pass both checks for the command to be issued.

However, each can be used independently. If you just have command security active, the only check performed is to determine whether the issuer of the command is authorized to issue the command. If authority is granted, the command will be issued. If you just have command resource security active then anyone can issue commands that do not affect resources, but if a resource is affected, a command resource security check will be performed to determine whether the issuer is allowed to issue that command for the named resource.

An example of how to use switch profiles follows.

Three WebSphere MQ subsystems have been defined called PROD, DEVT, and TEST. All the RACF classes have been defined and activated.

The three systems have different requirements:

- ▶ PROD system wants full security
- ▶ DEVT only wants connection and queue security active
- ▶ TEST does not want any security at all

So how do these requirements relate to switch profiles?

- ▶ PROD wants full security

For this you need to ensure that there are no PROD.NO switch profiles defined in the MQADMIN class

You can check this by using the RACF SEARCH command

```
SEARCH CLASS(MQADMIN) MASK(PROD.NO)
```

and checking the output.

- ▶ DEVT- requires connection and queue security only

This can be done by defining DEVT.NO switch profiles for each of the security types you *do not* want.

```
RDEFINE MQADMIN DEVT.NO.CMD.CHECKS
RDEFINE MQADMIN DEVT.NO.CMD.RESC.CHECKS
RDEFINE MQADMIN DEVT.NO.PROCESS.CHECKS
RDEFINE MQADMIN DEVT.NO.NLIST.CHECKS
RDEFINE MQADMIN DEVT.NO.CONTEXT.CHECKS
RDEFINE MQADMIN DEVT.NO.ALTERNATE.USER.CHECKS
```

You should also ensure there are no other DEVT.NO switch profiles in the MQADMIN class by using the **SEARCH** command as shown for PROD and checking that only the ones you want are defined.

- ▶ TEST requires no security at all

This is done by defining the NO.SUBSYS.SECURITY switch profile in the MQADMIN class.

```
RDEFINE MQADMIN TEST.NO.SUBSYS.SECURITY
```

This turns off security checking for that entire subsystem until such a time that the switch is reset.

4.5.3 WebSphere MQ for z/OS changes for V5.2

WebSphere MQ for z/OS V5.2 introduced the concept of shared queues and queue sharing groups using WebSphere MQ, a coupling facility (CF) and DB2.

Queue sharing groups

A queue manager can belong to one queue sharing group. The group a queue manager belongs to is defined in a new ZPARM. Queue managers in the same queue sharing group can use shared queues in that group.

The shared queue definitions are held in DB2, for recovery, and accessed from memory, for performance. The messages on shared queues are held in the CF.

Shared queues

Messages can be put on shared queues by any queue manager in the queue sharing group.

Messages can be got from shared queues by any queue manager in the queue sharing group.

Coupling facility (CF)

Messages are stored in CF list structures non-persistent messages in V5.2 and both persistent and non-persistent messages in V5.3. These have a limit of 63 kilobyte message length.

DB2 usage

DB2 is used to hold the shared WebSphere MQ object definitions, for example, queues and channels.

Figure 4-2 shows an overview of the concept of three queue managers in a queue sharing group, all accessing a CF for shared messages on shared queues and DB2 for shared definitions.

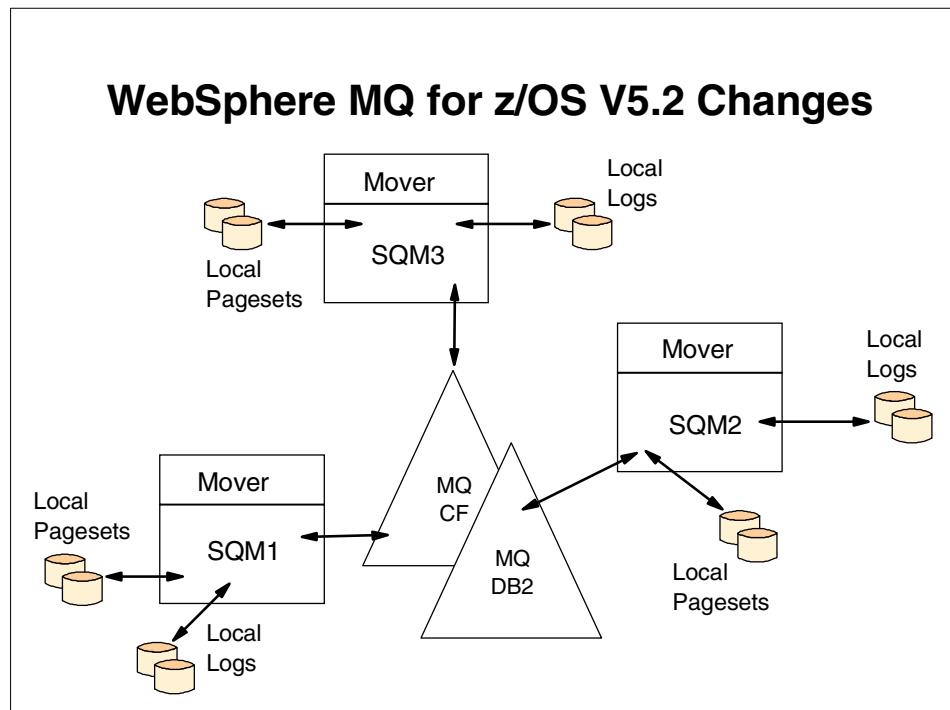


Figure 4-2 Concept of queue managers in a queue sharing group accessing a CF for shared messages

4.5.4 What has changed in WebSphere MQ for z/OS V5.2 security?

Use of DB2

Shared queue managers connecting to DB2 will have connection checking performed by DB2. Use existing security facilities in DB2 to set this up. The

shared queue manager address space user ID will be used for this connection check.

Use of coupling facility (CF)

In order for a shared queue manager to have access to structures in the CF, each queue manager address space user ID must have ALTER access to the relevant RESOURCE(IXLSTR.structure_name) profiles in the CLASS(FACILITY).

Queue sharing groups (QSG)

At shared queue manager startup, the queue manager will attempt to connect to a Cross Coupling Facility (XCF) group for the queue sharing group. The IXCCREAT to the group is subject to security checks, and these will ensure that the shared queue manager is allowed to join the group.

Use existing security facilities in these areas.

Queue managers

From WebSphere MQ for z/OS V5.2, a queue manager can be known as a local queue manager or a shared queue manager, depending upon whether or not it belonged to a queue sharing group.

Local queue managers

There are no major changes except for reslevel audit and refresh security.

Shared queue manager

New features are as follows:

- ▶ Queue sharing group high level qualifiers

Queue sharing group profiles will use the queue sharing group ID as the hlq instead of a ssid. For example:

```
qsg.resourcename
```

So, one queue sharing group profile could be used:

- To set up a single security switch profile for all queue managers within the queue sharing group, for example:

```
- qsg.NO.COMD.CHECKS
```

This would turn off command security checking on all queue managers in the queue sharing group.

- To control access to a single shared queue from any user on any shared queue manager within the queue sharing group. For example:

```
qsg.SHARED.QUEUE.ALL
```

protects a shared queue called SHARED.QUEUE.ALL from access by any user on any queue manager within the queue sharing group.

- To control access to a local queue of the same name on any of the shared queue managers within the queue sharing group where local checking is not active or where there is no local profile, for example:

```
h1q.LOCAL.QUEUE.EVERYWHERE
```

protects all local queues called LOCAL.QUEUE.EVERYWHERE defined on queue managers within the queue sharing group.

► New switches and new switch profiles

- Queue manager checks switch, controlled by the profile

```
h1q.NO.QMGR.CHECKS
```

This is used to determine whether or not security checks can use profiles with a hlq of ssid.

- Queue sharing group checks switch, controlled by the profile

```
h1q.NO.QSG.CHECKS
```

This is used to determine whether or not security checks can use profiles with an hlq of qsg.

You cannot set both queue manager and queue sharing group to OFF together; if you try this, you will have both queue manager and queue sharing group security activated.

The default is for both to be active and for ssid qualified profiles to be looked for first.

However, this means that you can be in a situation where, for example, a profile:

```
QSG1.NO.CMD.CHECKS
```

has turned OFF command security on all queue managers within the queue sharing group, but one queue manager wants command security ON.

So for every hlq.NO profile, there is now an equivalent ssid.YES profile which will override a qsg.NO profile for a given subsystem if set up correctly.

► New rules associated with the new features and how they affect the remainder of security processing.

- The default is to check ssid profiles before queue sharing group profiles
- ssid.YES profiles override qsg.NO profiles
- If queue manager checks switch is ON / queue sharing group checks switch is OFF this means that *only* profiles with a hlq of ssid will be used.

- If queue sharing group checks switch is ON / queue manager checks switch is OFF this means that *only* profiles with hlq of qsg will be used.
- If both queue manager and queue sharing group switches are ON then an hlq of ssid will be used first and if it is not found then a hlq of qsg will be used on the security check.
- Once the queue manager and queue sharing group switches have been determined then the remaining switch profiles are checked using the queue manager/queue sharing group rules.
- Once the shared queue manager is up and running, all security checks performed are governed by the following:
 - Whether the switch setting for that type of security is set to ON or OFF
 - Whether we are doing queue manager only, queue sharing group only, or both the queue manager/queue sharing group level checking
 - The Reslevel setting if it applies

If this queue manager is known as a local queue manager it has only to decide whether or not it wants security turned on on this queue manager and if so, at what level.

If the queue manager is known as a shared queue manager it then has a further three options to choose from once it has decided it want security turned on. Does this queue manager want security checking to take place at a queue manager level, a queue sharing group level or a combination of both queue manager and queue sharing group levels?

The diagrams in Figure 4-3, Figure 4-4 on page 73 and Figure 4-5 on page 74 show the order in subsystem security, queue manager level security and queue sharing group level security switch profiles that are checked during queue manager startup and security initialization to determine the level of checking that will occur.

This figure shows the order of subsystem security and switch profiles checked.

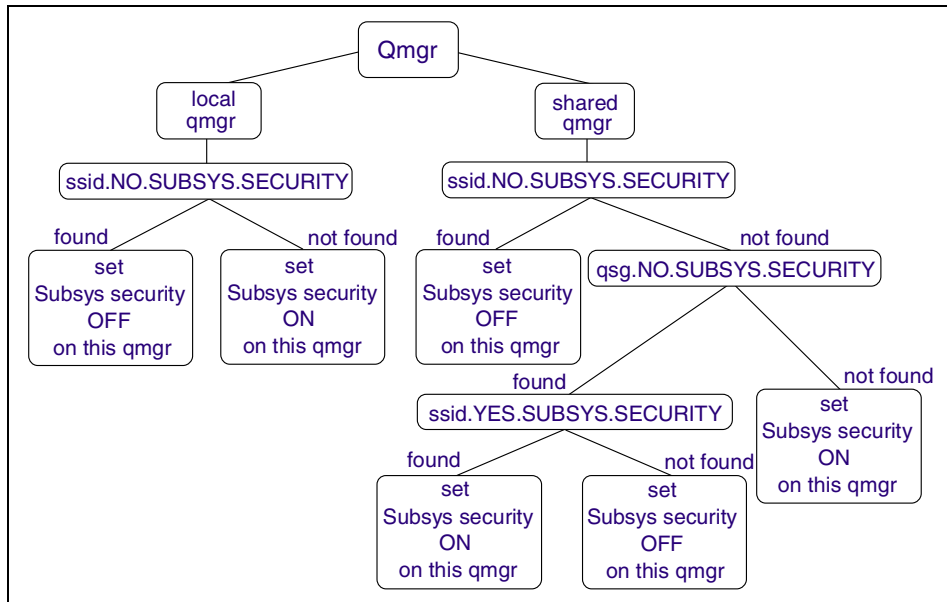


Figure 4-3 Subsystem security switch profile checking

This figure shows the queue manager level security and switch profiles checked.

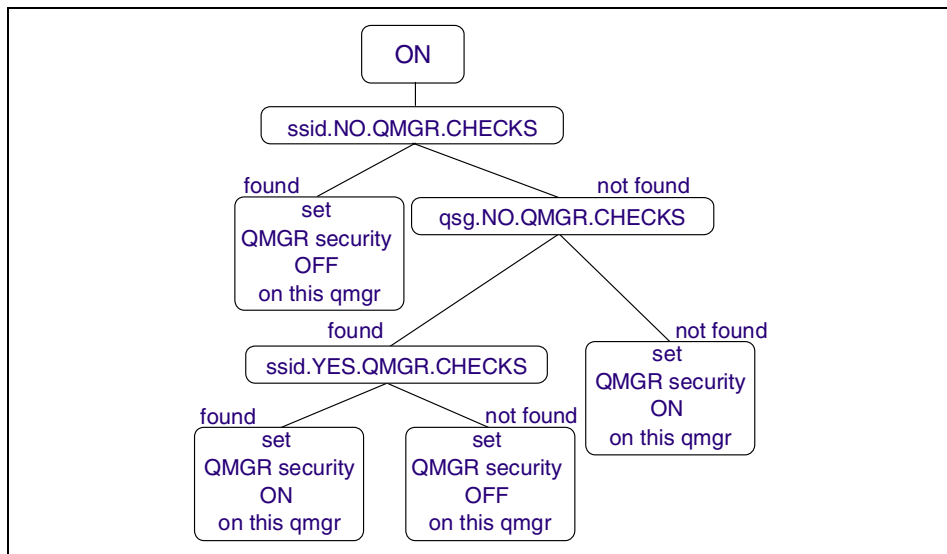


Figure 4-4 Queue manager level security switch profile checking

This figure shows the queue sharing group level security and switch profiles checked.

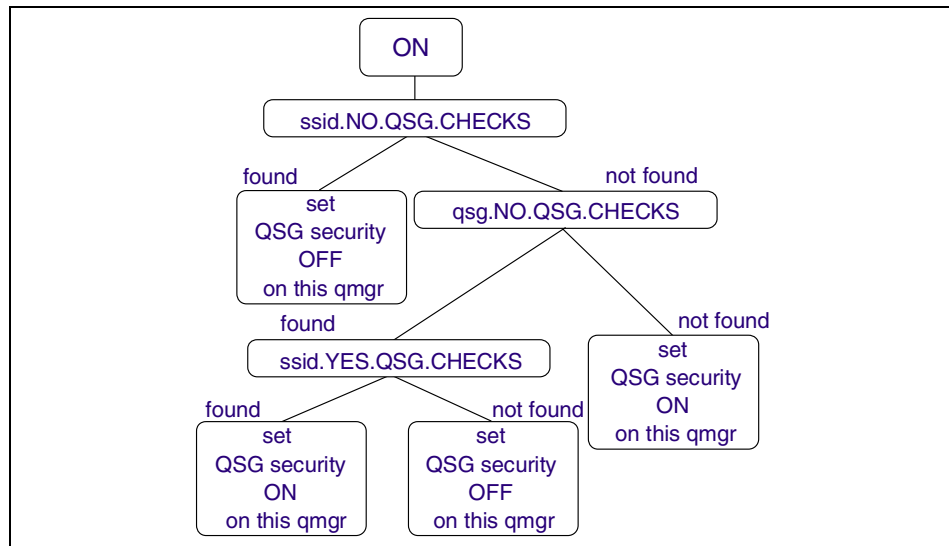


Figure 4-5 Queue sharing group level security switch profile checking

Once the subsystem, queue manager and queue sharing group switches have been set up, the results of those checks are used to determine how much checking needs to be done for the remainder of the switches.

If both queue manager and queue sharing group switches are active then we follow the process shown for the subsystem, queue manager and queue sharing group level checking to determine how to set the remaining switches.

If either queue manager only or queue sharing group only checking is active then we only look for the switch profile with the relevant high level qualifier.

For example, if queue sharing group only checks are active, for the remainder of the switches we only check for

`qsg.NO.'type'.checks profiles,`

where 'type' is substituted with the relevant security switch type (the type could be CONNECT, QUEUE, NLIST, etc.).

- Improved security switch messages: there is a new format for most security messages to help you see what security is set up and why.

Security messages are now written to the log at queue manager startup.

Example 4-1 Examples of security messages issued at startup; initialization of a queue manager

```
13.35.26 STC01960 CSQH024I !MQ19 CSQHINSQ SUBSYSTEM security switch set
        profile 'SQ05.NO.SUBSYS.SECURITY' not found
13.35.26 STC01960 CSQH022I !MQ19 CSQHINSQ QMGR security switch set ON,
        profile 'MQ19.YES.QMGR.CHECKS' found
13.35.26 STC01960 CSQH021I !MQ19 CSQHINSQ QSG security switch set OFF,
        profile 'SQ05.NO.QSG.CHECKS' found
13.35.26 STC01960 CSQH021I !MQ19 CSQHIS1C CONNECTION security switch set OFF,
        profile 'MQ19.NO.CONNECT.CHECKS' found
13.35.26 STC01960 CSQH024I !MQ19 CSQHIS1C COMMAND security switch set ON,
        profile 'MQ19.NO.CMD.CHECKS' not found
13.35.26 STC01960 CSQH021I !MQ19 CSQHIS1C CONTEXT security switch set OFF,
        profile 'MQ19.NO.CONTEXT.CHECKS' found
13.35.26 STC01960 CSQH024I !MQ19 CSQHIS1C ALTERNATE USER security switch set
        ON, profile 'MQ19.NO.ALTERNATE.USER.CHECKS' not found
13.35.26 STC01960 CSQH021I !MQ19 CSQHIS1C COMMAND RESOURCES security switch
        set OFF, profile 'MQ19.NO.CMD.RESC.CHECKS' found
13.35.26 STC01960 CSQH024I !MQ19 CSQHIS1C PROCESS security switch set ON
        profile 'MQ19.NO.PROCESS.CHECKS' not found
13.35.26 STC01960 CSQH024I !MQ19 CSQHIS1C NAMELIST security switch set ON,
        profile 'MQ19.NO.NLIST.CHECKS' not found
13.35.26 STC01960 CSQH024I !MQ19 CSQHIS1C QUEUE security switch set ON,
        profile 'MQ19.NO.QUEUE.CHECKS' not found
```

Messages of the same format are written to the log during refresh security processing.

The messages written to the log for a display security command are a shortened version of the messages, because they are also used by the Ops and Control panels.

Example 4-2 Shortened version of the security messages

```
13.36.48 STC01960 CSQH015I !MQ19 Security timeout = 54 minutes
13.36.48 STC01960 CSQH016I !MQ19 Security interval = 12 minutes
13.36.48 STC01960 CSQH030I !MQ19 Security switches ...
13.36.48 STC01960 CSQH034I !MQ19 SUBSYSTEM: ON, 'SQ05.NO.SUBSYS.SECURITY' not
found
13.36.48 STC01960 CSQH036I !MQ19 QMGR: ON, 'SQ05.NO.QMGR.CHECKS' overridden
13.36.48 STC01960 CSQH036I !MQ19 QSG: ON, 'SQ05.NO.QSG.CHECKS' overridden
13.36.48 STC01960 CSQH031I !MQ19 CONNECTION: OFF, 'MQ19.NO.CONNECT.CHECKS'
found
13.36.48 STC01960 CSQH034I !MQ19 COMMAND: ON, 'SQ05.NO.CMD.CHECKS' not found
13.36.48 STC01960 CSQH031I !MQ19 CONTEXT: OFF, 'MQ19.NO.CONTEXT.CHECKS' found
13.36.48 STC01960 CSQH034I !MQ19 ALTERNATE USER: ON,
'SQ05.NO.ALTERNATE.USER.CHECKS' not found
13.36.48 STC01960 CSQH034I !MQ19 PROCESS: ON, 'SQ05.NO.PROCESS.CHECKS' not
found
```

```
13.36.48 STC01960 CSQH034I !MQ19 NAMELIST: ON, 'SQ05.NO.NLIST.CHECKS' not
found
13.36.48 STC01960 CSQH034I !MQ19 QUEUE: ON, 'SQ05.NO.QUEUE.CHECKS' not found
13.36.48 STC01960 CSQH031I !MQ19 COMMAND RESOURCES: OFF,
'MQ19.NO.CMD.RESC.CHECKS' found

13.36.48 STC01960 CSQ9022I !MQ19 CSQHPDTC ' DISPLAY SECURITY' NORMAL
COMPLETION
```

- ▶ Intra Group Queueing (IGQ).

See “Channels and Intra-Group Queueing (IGQ)” on page 81 for further details.

4.5.5 Connection security

If connection security is active, you must define profiles in the MQCONN class and permit the necessary user IDs' access to those profiles so they can connect to the queue managers. The profiles needed in the MQCONN class are:

```
hlq.BATCH
hlq.CICS
hlq.IMS
hlq.CHIN
```

There is one profile per adapter type, which provides granular control for each environment.

READ access is required by the WebSphere MQ adapter user ID connecting to the queue manager in each environment.

Queue manager only checking:

If you have a hlq of a queue managername, then the profile controls access to the related queue manager for that connection type.

Queue sharing group only checking:

If you have a hlq of a queue sharing group name, then the profile controls access to all queue managers within the queue sharing group for that connection type.

4.5.6 API security

API security is controlled by several profiles in different RACF classes:

- ▶ MQQUEUE class

```
hlq.queueName
```


- ▶ MQADMIN class
 - hlq.CONTEXT.queueName
 - hlq.ALTERNATE.USER.alternateuserid
- ▶ MQPROC class
 - hlq.processName
- ▶ MQNLIST
 - hlq.nameListName

Access control checks for the WebSphere MQ API are performed when a queue is opened (MQOPEN or MQPUT1) and (for permanent dynamic queues) when a queue is closed (and deleted). Because of the different ways in which a queue may be opened, there are different access control checks and different access levels which are required. WebSphere MQ provides five different profiles, depending on the object being opened and the access required. There may also be several checks performed if a queue is being opened in a specific way; if a dynamic queue is being created, alternate user IDs are to be used and MQMD context fields are to be accessed. The total number of checks that might be made is eight if the RESLEVEL profile requires that two user IDs be checked.

For model queues, there are a number of security related aspects to consider: Two checks are performed:

1. Authorization to access the model queue
2. Authorization to access the dynamic queue to which model queue resolves

Dynamic queues and generic names can be created for those. There are several things to consider here too, but it is especially important to ensure that the dynamic queue names have appropriate profiles defined for them. This will, most likely, involve the use of a generic profile.

Resource security checking is performed during closure of permanent dynamic queues. This is performed if an application opens a permanent dynamic queue that it did not create and then attempts to delete it; an additional security check is carried out to see if it has authority to do so.

Note: The named queue only is checked.

- ▶ In the MQQUEUE class, the profiles are of the type:
 - hlq.queueName
 Required access to the profile is dependent upon the options specified on:
 - MQOPEN
 - MQPUT1

Inquire and Browse require READ access, Set requires ALTER access, all others require UPDATE access.

Access granularity is different from non-z/OS systems; MQGET is the same as MQPUT. You can get around this by the use of alias queues to distinguish between putters and getters.

MQOPEN for dynamic queues requires access to multiple profiles:

- Model queue profile
- Dynamic queue profile

MQCLOSE checking for permanent dynamic queues requires close delete access to the hlq.queueName profile.

► MQADMIN class

hlq.CONTEXT.*

These profiles control access to MQMD context fields; the required access to the profile is dependent upon the type of action to be performed on the context fields. There should be one profile per queue, per queue manager or queue sharing group.

Some applications require access to the MQMD context fields. These fields are protected by the context security profile and, for some types of access, by the queue profile. The type of access required to the context fields varies and so the access required to the context security profile will vary accordingly.

These fields include the user identifier; this is typically used to determine the alternate user ID that is to be used for processing the message. The

hlq.ALTERNATE.USER.alternateuserid

profile controls the use of an alternate user ID. To use an alternate user ID, you require UPDATE access to the appropriate profile. There should be one profile per alternate user ID, per queue manager or queue sharing group.

Alternate user ID profiles are used to control who is allowed to perform work for another user, under that alternate user's authority. For example, a server may be receiving work from lots of different places and needs to be able to put those messages on behalf of the users who sent them. In order to do this, the server would have to have the correct authority granted against the appropriate alternate user ID profile. Thus, if user ID SERVER54 needs to do work on behalf of USER12, SERVER54 needs UPDATE access to the hlq.ALTERNATE.USER.USER12 profile.

Note that WebSphere MQ user IDs can be 12 characters long and all 12 characters may be used on the profile for alternate user authority. Even so, only the first eight characters of the user ID are used for any security check performed.

► MQPROC class

hlq.processname

READ access is required by the user ID(s).

▶ MQNLIST class

hlq.namelistname

READ access is required by the user ID(s).

Processes and Namelists are opened for inquiry only.

There is no checking for the queue manager object.

4.5.7 Command and command resource security

▶ MQCMDS class

hlq.verb.pkw

For example:

hlq.DEFINE.QLOCAL

The access required to the profile is dependent upon the verb. This allows completely granular control of WebSphere MQ commands, much more than with non-z/OS systems.

▶ MQADMIN class

hlq.type.resourcename

For example:

hlq.QUEUE.queueename

The access required to the profile is dependent upon the verb; it is usually ALTER or CONTROL.

These two profiles allow very granular control of the WebSphere MQ commands. There is a separate profile for each WebSphere MQ command (the verb) and target (the primary keyword), allowing each command to be controlled individually. Thus, a particular user ID may be able to define qlocals but not define qremotes or may be able to display queues but not define them. It is also possible to control access to the resources accessed by these commands. Thus, a user may be authorized to use the ALTER QLOCAL command but not alter a specified queue.

Clearly, there is a price to pay with respect to this control; if this type of granular control is required then many profiles may need to be defined to facilitate this access control.

See Chapter 13 of the *WebSphere MQ for z/OS System Setup Guide*, SC34-6052 for a table showing the profiles and access required for each profile.

4.5.8 User ID and reslevel

All access control checks are based upon the availability of one or more user IDs. The user IDs used depend upon the environment/adaptor in use.

For all environments, the address space user ID is usually used. This address space user ID may not be available if the address space is a started procedure. In this case, a user ID is obtained from the z/OS started procedures table.

Whether or not a second user ID is used in addition to the address space user ID depends upon the reslevel profile and what access the connecting address space user ID has to that profile. The

hlq.RESLEVEL

profile is defined in the MQADMIN class and controls the number of user IDs checked.

Batch

For batch address spaces, there is only one user ID available. For CICS and IMS address spaces, where many independent transactions are running, a second user ID is available and for the Mover and IGQ, there is also a second user ID available.

For Batch, the user ID is the address space user ID or the TSO user ID

CICS

For CICS, the user IDs are the address space user ID and the transaction user ID

IMS

For IMS, the user IDs are the address space user ID and a “second” user ID which varies according to the type of IMS dependent region in use, as follows: when IMS transactions connect to WebSphere MQ, there are (as with CICS on z/OS) two user IDs used for controlling access to WebSphere MQ resources.

These are:

1. The address space user ID, for either the IMS Control region or IMS Dependent region
2. One of the user IDs associated with the IMS transaction
 - LTERM name
 - PSBNAME

The choice of which of these to use is driven by the type of dependent region, according to Table 4-1.

Table 4-1 Determining second IMS user ID

Type of dependent region	Hierarchy for determining 2nd user ID
1.BMP message driven and successful GET UNIQUE issued 2.IFP and GET UNIQUE issued 3.MPP	1.User ID associated with the IMS transaction if the user ID signed on or 2.LTERM name if available or 3.PSBNAME
1.BMP message driven and successful GET UNIQUE not issued 2.BMP not message driven 3.IFP and GETUNIQUE not issued	1.User ID associated with the IMS Dependent region address space if this is not blanks/zeroes or 2.PSBNAME

Channels and Intra-Group Queuing (IGQ)

For the z/OS Mover, the user IDs are the channel user ID and the MCA user ID or alternate user ID; which user IDs are used and where they are obtained depends upon the channel type, channel definition and the communications protocol used.

IGQ, shown in Figure 4-6 on page 82, is a way for less expensive, small, non-persistent messages to be transferred between queue managers within a queue sharing group without having to define channels between the queue managers. When applications open and put messages to remote queues, the local queue manager determines whether intra-group queuing should be used for message transfer.

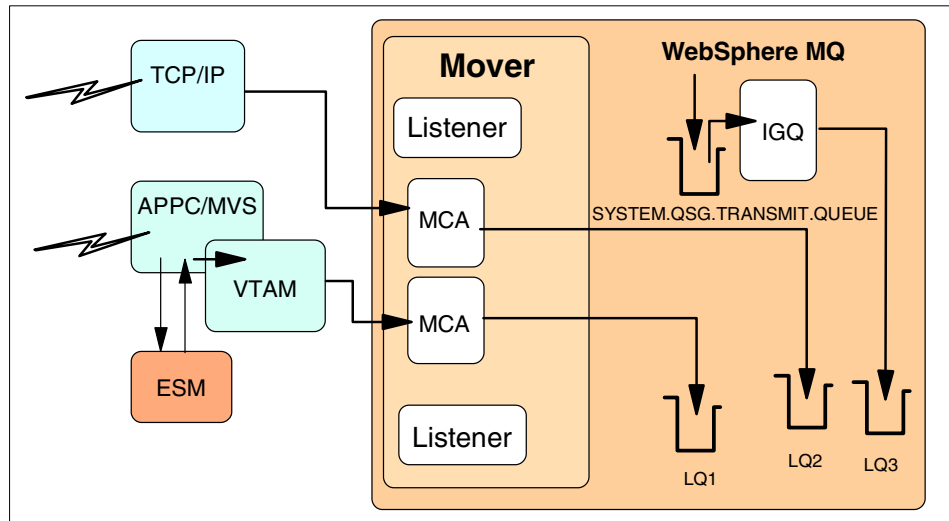


Figure 4-6 Channel and IGQ security

The user IDs that may be used for security checking are dependent upon environment and setup:

TCP/IP	Receiving channel initiator user ID or SSL obtained user ID, MCA user ID, ALT user ID
APPC	Sending channel initiator user ID, receiving channel initiator user ID, MCA user ID, ALT user ID
IGQ	IGQ user ID, sending queue manager user ID, ALT user ID

Profile name	PUTAUT option specified on receiver or requester channel							
	DEF		CTX		ONLYMCA		ALTMCA	
	1 check	2 checks	1 check	2 checks	1 check	2 checks	1 check	2 checks
ssid.Alternate.User.userid	---	---	CHL	CHL+MCA	---	---	MCA	MCA
ssid.Context.queueName	CHL	CHL+MCA	CHL	CHL+MCA	MCA	MCA	MCA	MCA
ssid.resourcename	CHL	CHL+MCA	CHL	CHL+ALT	MCA	MCA	MCA	MCA+ALT
Key: ALT Alternate User ID CHL Channel Userid MCA MCA user ID								

Figure 4-7 When each channel security related user ID is checked

Receiving channels using TCP/IP

► MCA user ID (MCA):

The user ID specified for the MCAUSER channel attribute at the receiver side; if blank, the channel initiator address space user ID of the receiver or requestor side.

► Channel user ID (CHL):

On TCP/IP, security is not supported by the communication system for the channel. If the Secure Sockets Layer (SSL) is being used and a digital certificate has been flowed from the partner, the user ID associated with this certificate (if installed), or the user ID associated with a matching filter found by using RACF's certificate name filtering (CNF), is used. If no associated user ID is found or if SSL is not being used, the user ID of the channel initiator address space of the receiver or requestor end is used as the channel ID on channels defined with the PUTAUT parameter set to DEF or CTX.

Note: The use of RACF's CNF allows you to assign the same RACF user ID to multiple remote users, for example all the users in the same organizational unit, who would naturally have the same authority. This means that the server does not have to have a copy of the certificate of every possible remote end user across the world; this greatly simplifies certificate management and distribution.

If the PUTAUT parameter is set to ONLYMCA or ALTMCA for the channel, the channel user ID is ignored and the MCA user ID of the receiver or requestor is used. This also applies to TCP/IP channels using SSL.

- ▶ Alternate user ID (ALT):

The user ID specified in the useridentifier field in the message descriptor of the message.

Receiving channels using APPC (LU6.2)

- ▶ MCA user ID (MCA):

The user ID specified for the MCAUSER channel attribute at the receiver; if blank, the channel initiator address space user ID of the receiver or requester side.

- ▶ Channel user ID (CHL):

Requester/server channels: if the channel is started from the requester, there is no opportunity to receive a network user ID (channel user ID). Because of this, the user ID of the channel initiator address space of the receiver or requestor end is used as the channel ID on channels defined with the PUTAUT parameter set to DEF or CTX.

If the PUTAUT parameter is set to ONLYMCA or ALTMCA for the channel, the channel user ID is ignored and the MCA user ID of the receiver or requester is used.

Other channel types: if PUTAUT is set to DEF or CTX on the receiver or requester channels, the channel user ID is the user ID received from the communications system when the channel is initiated. If the sending channel is z/OS, the channel user ID received is the channel initiator address space user ID of the sender. If the sending channel is on a different platform, the channel user ID received is typically provided by the USERID parameter of the channel definition.

If a user ID received is blank, or no user ID is received, then a channel user ID of blank is used.

- ▶ Alternate user ID (ALT):

The user ID specified in the useridentifier field in the message descriptor of the message

Client MQI requests

Profile name	PUTAUT option specified on receiver or requester channel							
	DEF				ONLYMCA			
	Alternate userid specified on open ?							
	NO		YES		NO		YES	
	1 check	2 checks	1 check	2 checks	1 check	2 checks	1 check	2 checks
ssid.Alternate.User.userid	---	---	CHL	CHL+MCA	---	---	MCA	MCA
ssid.Context.queue name	CHL	CHL+MCA	CHL	CHL+MCA	MCA	MCA	MCA	MCA
ssid.resourcename	CHL	CHL+MCA	CHL	CHL+ALT	MCA	MCA	MCA	MCA+ALT
Key: ALT Alternate User ID CHL Channel Userid MCA MCA user ID								

Figure 4-8 User IDs used for client channels

For server connection channels, the user ID received from the client is used if the MCAUSER attribute is blank. However, for the clients that can use the MQ_USER_ID environment variable to supply the user ID, it is possible that no environment variable has been set. In this case, the user ID that started the server channel is used. This is the user ID assigned to the channel initiator started task in the z/OS started procedures table.

For Client MQOPEN and MQPUT1 requests, the following rules determine which profiles are checked.

- ▶ If the request specifies alternate user authority, a check is made against the hlq.ALTERNATE.USER.user ID profile.
- ▶ If the request specifies context authority, a check is made against the hlq.CONTEXT.queue name profile.
- ▶ For all MQOPENS and MQPUT1 requests, a check is made against the hlq.resourcename profile.

IGQ

User IDs checked for Intra-Group queueing (IGQ)								
Profile name	IGQAUT option specified on receiving queue manager							
	DEF		CTX		ONLYIGQ		ALTIGQ	
	1 check	2 checks	1 check	2 checks	1 check	2 checks	1 check	2 checks
ssid.Alternate.User.userid	---	---	SND	SND+IGQ	---	---	IGQ	IGQ
ssid.Context.queueName	SND	SND+IGQ	SND	SND+IGQ	IGQ	IGQ	IGQ	IGQ
ssid.resourcename	SND	SND+IGQ	SND	SND+ALT	IGQ	IGQ	IGQ	IGQ+ALT
Key: ALT Alternate User ID IGQ IGQ user ID SND Sending Queue manager user ID								

Figure 4-9 User IDs for IGQ

► Intra-Group Queueing user ID (IGQ):

The user ID determined by the IGQUSER attribute of the receiving queue manager.

If this is set to blank, the user ID of the receiving queue manager is used. However, because the receiving queue manager has authority to access all queues defined to it, security checks are not performed from the receiving queue managers user ID.

If two user IDs are to be checked and one of the user IDs is that of the receiving queue manager, security checks will take place for the other user ID only. This can occur when IGQAUT is set to DEF, CTX or ALTIGQ.

Important: If only one user ID is to be checked and the user ID is that of the receiving queue manager, no security checks will take place. This can occur when IGQAUT is set to ONLYIGQ or ALTIGQ.

If two user IDs are to be checked and both user IDs are that of the receiving queue manager, no security checks will take place. This can occur when IGQAUT is set to ONLYIGQ.

► Sending queue manager user ID (SND):

The user ID of the queue manager within the queue-sharing group that put the message on to the SYSTEM.QSG.TRANSMIT.QUEUE.

- ▶ Alternate user ID (ALT):

The user ID specified in the `userid` field in the message descriptor of the message.

4.5.9 Security commands

There are four WebSphere MQ for z/OS commands that help you to administer security for your queue manager. The introduction of the `CMDSCOPE` keyword V5.2 provides the capability to issue the commands on one queue manager in the queue sharing group and have them distributed to the other queue managers within the same group to be processed as if they had been issued on that system.

- ▶ `DISPLAY SECURITY ALL|INTERVAL|SWITCHES|TIMEOUT`

This allows you to see what security is active on your queue manager and how frequently the internal clearout of security information held by the queue manager is performed.

- ▶ `REFRESH SECURITY(*|MQADMIN|MQQUEUE|MQPROC|MQNLIST)`

This command allows you to change your queue manager's security setup without bringing it down. There will be a performance impact while this command is being processed.

From version 5.2 of WebSphere MQ for z/OS on, you will have to perform the RACF commands

```
SETROPTS RACLIST(classname) REFRESH
SETROPTS GENERIC(classname) REFRESH
```

before you perform the WebSphere MQ Refresh security command; this is needed in order to refresh the information we now store in the RACF dataspace rather than the queue manager address space.

- ▶ `RVERIFY SECURITY(userid,userid...)`

This allows you to change particular users access levels whilst the system is up and running; once the change has been made in RACF, you need to issue this command, specifying each user ID that has had its authority changed.

- ▶ `ALTER SECURITY INTERVAL() TIMEOUT()`

This allows you to alter how frequently the internal clearout occurs and the period of time that information is allowed to remain unused in the Queue Manager.

4.6 WebSphere MQ security for OS/400, AIX and Windows

In WebSphere MQ for UNIX (IBM AIX, Solaris, HP-UX), OS/400 and Windows, there are methods of providing security.

- ▶ The object authority manager (OAM)
Checks access authority to WebSphere MQ objects such as queues and queue managers.
- ▶ DCE security (channel exits)
Channel exits that use the DCE Generic Security Service (GSS) are provided by WebSphere MQ.
- ▶ Channel security using SSL
Facilitates encryption and authentication of data, maintaining its integrity. It is available to both client/server channels and to queue manager/queue manager channels (including the cluster variants). See Chapter 10, “Architectural vulnerabilities” on page 177 for a detailed explanation.

4.6.1 WebSphere MQ security (OAM)

WebSphere MQ uses the security services provided by the underlying operating system.

OAM is part of WebSphere MQ for AIX, OS/400 and Windows.

It checks authorization for MQI calls used and commands that are enabled by default.

Access control list (ACL)

The OAM validates user access to WebSphere MQ objects by checking the ACL, using operating system group IDs to which the user belongs. ACL is stored for each object on a local queue called SYSTEM.AUTH.DATA.QUEUE. WebSphere MQ supplies commands to manage the ACL.

Principal

The OAM uses the term *principal* for WebSphere MQ users (*principal* is almost the same as *user*). When an application first connects to a queue manager, it checks the WebSphere MQ ACL for the user ID that is running the application and its group. A principal can belong to more than one group (group set) and the aggregate of all authorities is granted to each group in its group set.

If you change the membership of the group, you need to restart the queue manager or issue the **MQSC REFRESH SECURITY** command.

Identify the user ID

This section explains how to identify user IDs on the systems listed below.

AIX systems

All ACLs are based on group IDs. When a user is granted access to a particular object, it is the primary group ID of the user that is granted access and not the individual user ID. For example, if you add PUT authority to user1 whose primary group is *staff*, all users that belong to staff will get PUT authority. All unknown users are assigned to the default user group *nobody* and no authorizations are given by default.

By default, AIX user's primary group is set to *staff*, so consider the consequences for the user's primary group when granting authorization.

OS/400

All ACLs are based on user IDs since OS/400 treats groups the same way it treats users. By default, the security officer and the administrator have full access to all objects and one of these must set authorizations for additional users.

Windows systems

All ACLs are based on user IDs and group IDs. ACL checks are the same as in UNIX but they appear for individual user IDs on each object. The group name always points to the *local group*.

Authorization for each object

You can set various authorization for each object.

Table 4-2 Authorizations for objects

Authorization	Queue	Process	QMgr
all, alladm, allmqi, chg, crt, dlt, dsp, inq, set	x	x	x
browse, clr, put, get, passall, passid	x	N/A	N/A
setall, setid	x	N/A	x
altusr, connect	N/A	N/A	x

OAM commands

OAM administration commands are as follows:

- ▶ dspmqaut (display authority)
- ▶ dmpmqaut (dump authority)
- ▶ setmqaut (grant/revoke authority)

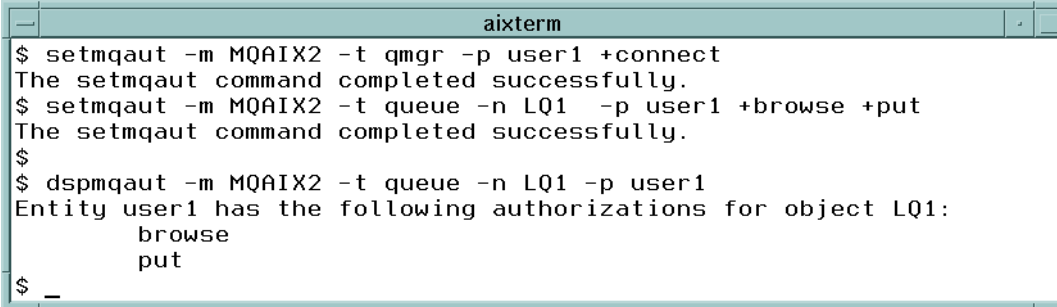
Command samples

- ▶ Grant the user called user1 authorization to connect to queue manager MQAIX2 and browse and put for the queue called LQ1.

```
setmqaut -m MQAIX2 -t qmgr -p user +connect  
setmqaut -m MQAIX2 -t queue -p LQ1 -p user1 +browse +put
```

- ▶ Display user1's authorization information relating to LQ1:

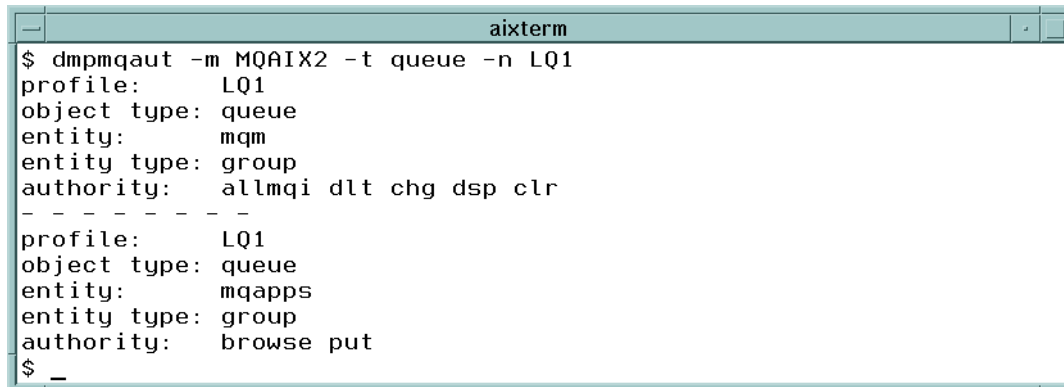
```
dspmqaut -m MQAIX2 -l queue -n LQ1 -p user1
```

The image shows a terminal window titled 'aixterm'. The terminal displays the following sequence of commands and their outputs:

```
$ setmqaut -m MQAIX2 -t qmgr -p user1 +connect  
The setmqaut command completed successfully.  
$ setmqaut -m MQAIX2 -t queue -n LQ1 -p user1 +browse +put  
The setmqaut command completed successfully.  
$  
$ dspmqaut -m MQAIX2 -t queue -n LQ1 -p user1  
Entity user1 has the following authorizations for object LQ1:  
    browse  
    put  
$ _
```

Figure 4-10 OAM command sample

- ▶ Display authorization information of LQ1:
dmpmqaut -m MQAIX2 -t queue -n LQ



```
aixterm
$ dmpmqaut -m MQAIX2 -t queue -n LQ1
profile:      LQ1
object type:  queue
entity:       mqm
entity type:  group
authority:    allmqi dlt chg dsp clr
- - - - -
profile:      LQ1
object type:  queue
entity:       mqapps
entity type:  group
authority:    browse put
$
```

Figure 4-11 OAM command sample 2

Generic profiles

With WebSphere MQ V5.3, you can use a generic profile to set up OAM. This helps you to manage many objects at once, rather than issue separate `setmqaut` commands against each individual object.

4.6.2 MQSeries administration wrapper (support pac MS0E)

You can define, delete and alter an object, such as a queue, a queue manager and a channel, by issuing a `runmqsc` command on UNIX (IBM AIX, Solaris, HP-UX) and Windows.

runmqsc command

Only the user that belongs to the mqm group or has execute permission can run this command. When you want to grant additional authority to another user, it is not necessary to make this user a member of the mqm group. Since members of the mqm group can perform all administration tasks, you must restrict access to this group.

For example, some users may only require access to display queue definitions but not to modify or delete them.

MQSeries administration wrapper

The support pac MQSeries administration wrapper (MS0E) is available from this Web link:

<http://www-3.ibm.com/software/ts/mqseries/txppacs/ms0e.htm>

If you install this support pac, users other than the mqm group can also execute WebSphere MQ management commands. The WebSphere MQ administration wrapper can be used to define layered administration roles. For example, some users may be allowed full control of WebSphere MQ channels but may not be allowed to modify queue definitions.

The support pac also creates an audit log of users using the administration commands.

This is a category 3 WebSphere MQ SupportPac™. If you encounter what you believe to be a defect with the SupportPac, you may report the problem using the same defect reporting channel you employ for the WebSphere MQ server product(s).

4.7 WebSphere MQ security (alternate user authority)

When an application opens a queue with an open option such as INPUT, OUTPUT, INQUIRE or SET, the queue manager checks whether the user ID has the full authority to access the queue.

By default, the user ID of an application issuing an **MQOPEN** is used to check the authorization.

We can create another user ID to be checked for authorization by using the alternate user ID function. Only an application can be used to verify this user ID and check for authorization using an alternate user ID in **MQOO** when the queue is opened.

This function is useful for a server application to check each user who puts a request message to the queue for authorization.

For example, there is an application that gets request messages from many users and forwards them to another queue. You can use each user ID that puts the request message to check the **PUT** authority by setting an alternate user ID as illustrated in the figure below.

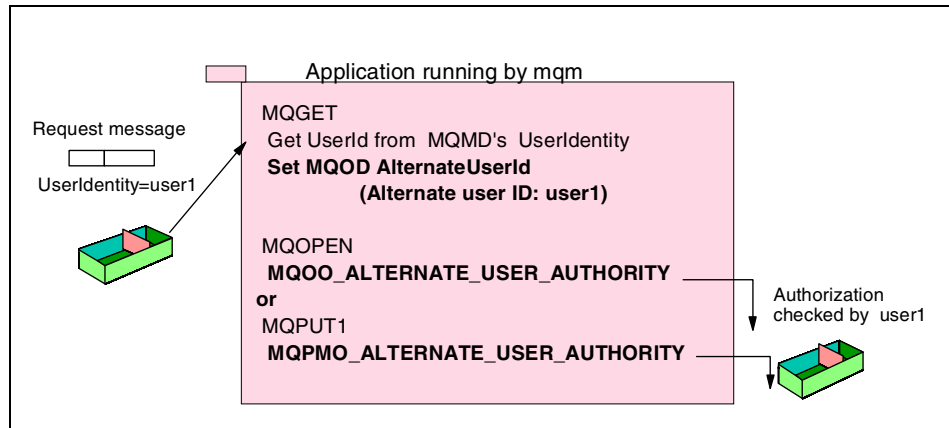


Figure 4-12 How to set an alternate user ID

4.8 WebSphere MQ channel security

This section describes channel security for WebSphere MQ.

4.8.1 Distributed queuing channel

This is a channel that connects two queue managers.

Connection security

When you start a channel, authorization is only required for local channels. Remote connections do not need any authorization.

Therefore, if someone knows your TCP/IP address or hostname and channel name, they can easily connect to your WebSphere MQ system. The port number that your listener listens on, which is specified in the conname field of the channel definition, generally defaults to 1414. This is a security exposure.

You can verify the partner connection as follows:

- ▶ Channel security exit
 - Set the security check program written by the user before the channel started.
- ▶ Channel SSL support
 - Filter the Distinguished Name (DN) in the certificate.

Establishing queue security

Tip: You should change the port number from the default 1414 for your connection security.

Once the channel is started, the authorization check is based on the PUTAUT attribute of the receiving channel when the message channel agent (MCA) writes the message into a local queue.

Table 4-3 *PUTAUT attribute of receiving channel*

PUTAUT I	User ID which used for authority check
DEF	Default user ID (running the MQ listener or inetd)
CTX	User ID contained in the MQMD of message

If you set PUTAUT=CTX and its user ID does not have enough authority, the message is put to a dead letter queue.

Cluster channel security

An advantage of a WebSphere MQ cluster is that a channel definition is created automatically. The disadvantage is that a channel which you do not want to connect to is also automatically created and connected.

To prevent selected queue managers from sending messages to your queue manager, we recommend that you define a channel exit on the CLUSRCVR channel.

4.8.2 WebSphere MQ client channel

Each WebSphere MQ Client program has its own channel. It is started when the program issues an **MQCONN** to connect to the queue manager on the remote machine and stop when **MQDISC** is issued.

The access control for an object is checked by user IDs, the same way as for local applications.

The user IDs used for WebSphere MQ client application are as follows:

- ▶ MCAUSER attribute of SVRCONN
- ▶ User ID running the application
- ▶ User ID running the listener or inetd

Usually, an application execution user ID is used, because the MCAUSER is set to blank by default. It is necessary to define all the user IDs connected to your system and to grant suitable authority to a server.

We recommend that the MCAUSER field be set up with a user ID and that suitable authority for this user be granted.

You can also use channel security exit and SSL support in the same way as you would on a distributed channel.

4.9 WebSphere MQ security (channel exits)

WebSphere MQ provides for channel security through various channel exit points to insert user-written exits. This is supported on all platforms.

Types of exits are:

- ▶ Security exit
- ▶ Message exit
- ▶ Send/receive exit
- ▶ Message-retry exit
- ▶ Channel auto-definition exit
- ▶ Transport-retry exit

The four major exits can be described as shown below.

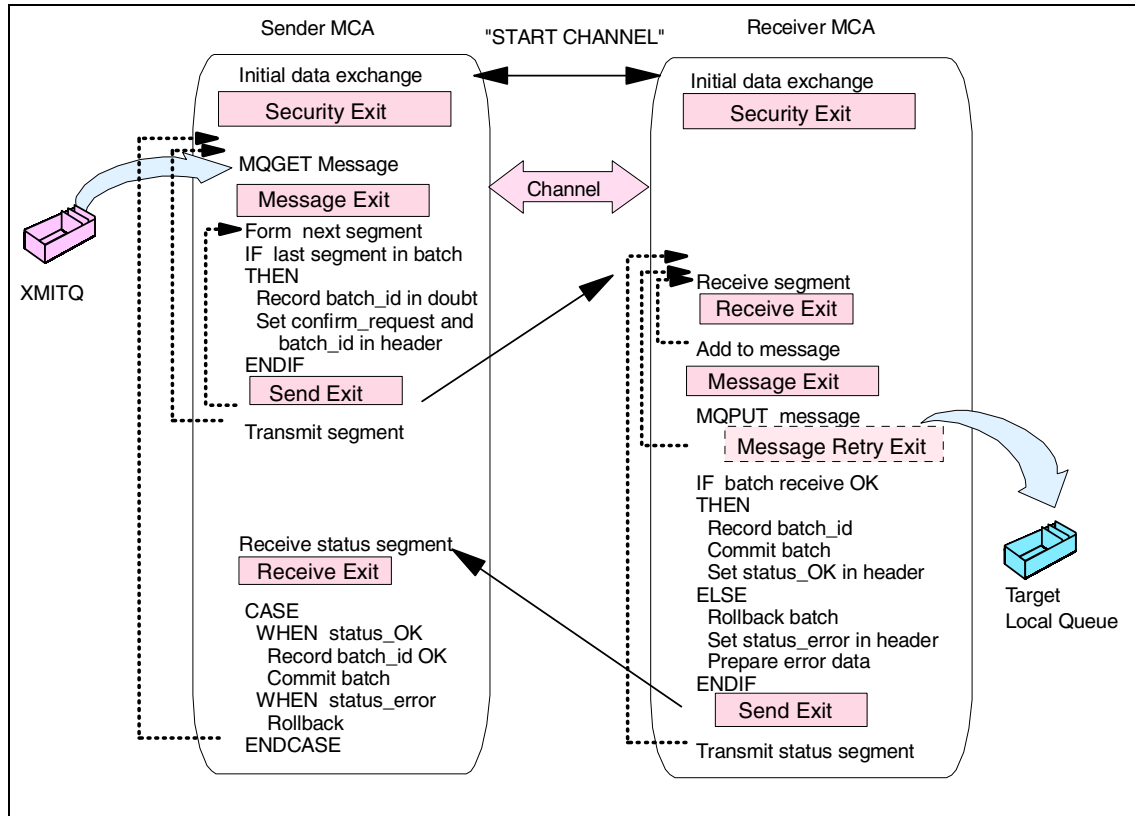


Figure 4-13 WebSphere MQ channel exit

Security exit

When a channel is started, the security exit is called after the initial channel data negotiation has completed. It is used for security checking, such as authentication of the partner. If authentication fails in a security exit, the channel is not started and the connection from WebSphere MQ client program fails.

Tip: With WebSphere MQ now supporting SSL, you can authenticate the partner by digital certificate and define the MQSSLPEER attribute to filter the partner by DN in the certificate. However, this filtering function by itself is not enough, so we recommend the use of a security exit for stricter user authentication.

Message exit

This is used for operations on a message. You can check the contents of the message and modify some fields or encrypt it prior to transmission.

Send/receive exit

The send exit and receive exit must be set in pairs; they can be used, for instance, for data compression/decompression.



IBM Tivoli Access Manager for Business Integration

IBM Tivoli Access Manager for Business Integration (AMBI) was developed to manage the security concerns raised by many WebSphere MQ deployment encounters, such as consistent enforcement of access control to WebSphere MQ objects and the encryption of data in transit between WebSphere MQ objects.

In this chapter, we revisit the basic WebSphere MQ security issues and mechanisms before showing how IBM Tivoli Access Manager for BI can resolve these issues. These sections reiterate information in other sections of this book. The final part of the chapter discusses when it would be appropriate to use IBM Tivoli Access Manager for BI rather than the native WebSphere MQ security.

5.1 Revisiting the WebSphere MQ security issues

In this section, we look at the basic WebSphere MQ data flow and identify the areas of concern from a security perspective. The aim of this section is to distill a set of issues IBM Tivoli Access Manager for BI can solve.

5.1.1 WebSphere MQ data flow

The standard WebSphere MQ transaction involves data being written (PUT) into a queue and retrieved (GET) from the queue. Most WebSphere MQ implementations involve data being transferred between applications and systems by this mechanism, or a combination of these transactions.

In Figure 5-1, we have two applications that need to transfer application data between two machines using WebSphere MQ.

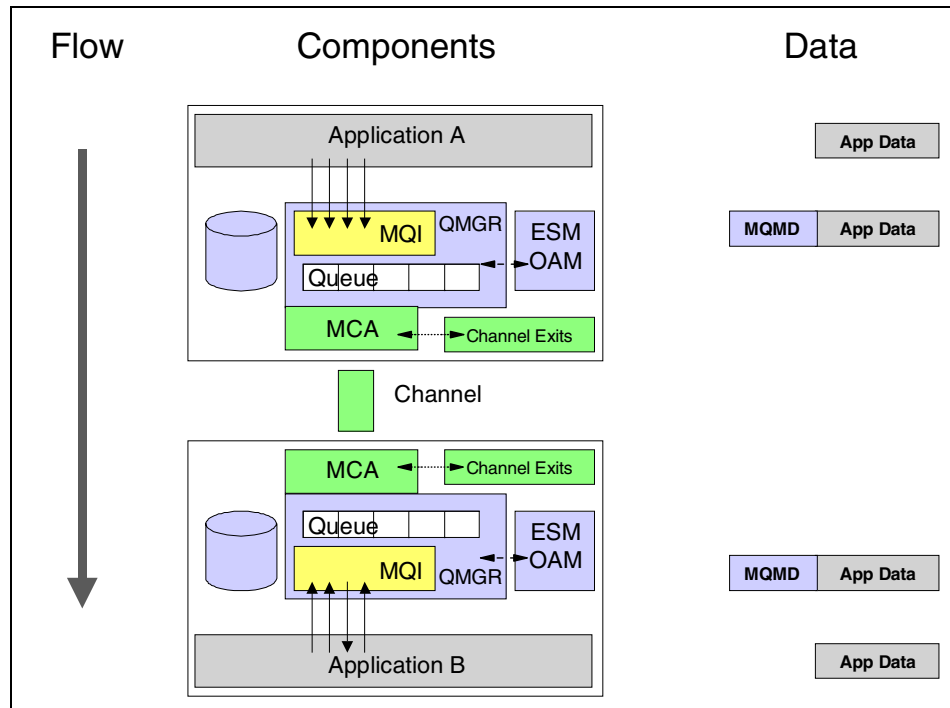


Figure 5-1 WebSphere MQ data flow

Application A formats the data and uses the WebSphere MQ API to connect to the queue manager (qmgr), open a queue for writing, put the application data in a queue, close the queue and disconnect from the queue manager. The API calls are handled by the Message Queue Interface (MQI) component. The put API call

will format the application data by putting a header (MQMD, or MQ Message Data).

Note: WebSphere MQ supports three different application programming interfaces, Java Message Service (JMS), Application Message Interface (AMI) and Message Queue Interface (MQI). On WebSphere MQ servers, the JMS and AMI interfaces are mapped to the MQI.

Access to the queue manager and queues is controlled by z/OS security Enterprise Security Manager (ESM) such as RACF or the Object Authority Manager (OAM) component of systems on other platforms, discussed later.

Up to this point, the application data has been held in storage. If the queue being written to is flagged as persistent, the message (application data plus the header) will be written to disk.

In our example, we are transferring application data between machines, so the queue that the application writes to is actually a queue on the other machine. The queue manager knows this, so the local queue that is written to locally is a transmit queue. The message channel agent (MCA) picks up messages from the transmit queue and sends them across the network via a channel.

The agent on the remote machine is listening for messages on the channel. It retrieves the messages and writes them to the appropriate queue, based on the information in the message header (MQMD). Again, if the queue is flagged as persistent, the message will be written out to disk.

Application B may be triggered by the arrival of messages on the queue, or it may periodically go and look for messages. It also uses the API to connect to the queue manager, open the queue, get the message(s), close the queue and disconnect from the queue manager. The data that is returned to the application is the application data without the message header.

This data flow presents a number of security issues which we discuss in the next section.

5.1.2 Native WebSphere MQ security issues

The key areas of concern from a security perspective are as follows.

- ▶ Access control

How do you control who has access to the queue managers, queues and other WebSphere MQ objects?

- ▶ Data security
How do you stop people from seeing message contents?
- ▶ Channel security
How do you stop people from tampering with message contents or sending “bogus” messages?
- ▶ Audit
How do you track security related events across a WebSphere MQ implementation?

These questions are discussed in the following sections.

Access control

With access control, we are concerned with controlling who has access to what WebSphere MQ objects, such as queue managers and queues. This is a significant issue in most WebSphere MQ implementations where many platforms are involved and different systems may have different levels of security applied. Loose access control on one platform, such as Windows or AIX, may expose a more secure system, such as z/OS.

Also, you may have different levels of security to be applied to application data flowing between systems. You need to be able to differentiate who or which applications can access unclassified data and who can access highly protected data.

On z/OS, WebSphere MQ uses an ESMr such as RACF (or equivalent access control mechanism, such as ACF/2 or Top Secret). On systems such as AIX and Windows, WebSphere MQ provides the OAM. This is a tool provided with native WebSphere MQ to control access to WebSphere MQ objects. The details of OAM are found in 4.6, “WebSphere MQ security for OS/400, AIX and Windows” on page 88.

One potentially significant issue for customers is that they must administer access control locally on every server. Queue creation and deletion are rather static actions, and do not need to be performed very often, but generally put and get permissions on queues are updated much more frequently as new applications are rolled out into production. This is more significant where access to queues is based on individual users rather than generic application logins.

With WebSphere MQ V5.3, you can use a generic profile to set up OAM. This helps you to manage many objects at once, rather than issuing separate commands against each individual object.

If access control definitions are being applied locally to a number of systems, you then have the issue of monitoring and enforcing an enterprise security policy. How do you guarantee that you are applying a consistent security policy to all systems when access is being defined manually on each system? An enterprise wide view of these security policies, with the ability for authorized administrators to remotely update them, may greatly improve efficiencies and assist audit compliance.

The other issue that may arise with the use of OAM in a large WebSphere MQ deployment is the management of identities. Depending on the platform, OAM may tie operating system accounts or account group membership to access control within WebSphere MQ. For example, when you add a user to a system, you may also need to add the user to OAM for WebSphere MQ access. When a user is removed or changes roles, you also need to change the user's OAM definitions.

So, by using OAM on systems such as AIX and Windows, you can apply granular access control to WebSphere MQ objects, but you introduce a number of other issues:

- ▶ Access control must be administered on each system, adding maintenance overhead and potential errors due to manual definition.
- ▶ With access control defined on each system individually, you have no centralized view or control of compliance to security policy, leading to possible audit exposures.
- ▶ As the access control definitions in OAM are tied to operating system identities, keeping these accounts synchronized with the access control may require some level of identity management.

These issues are more relevant to large WebSphere MQ deployments with many non-z/OS systems.

Data security

We are also concerned about data security, that is, being able to extract information by looking at messages. We have two areas of exposure with native WebSphere MQ: messages that are written to disk and messages that are sent across a network.

Messages are written to disk when a queue is flagged as persistent. The message data is in the format written by the application. Unless the application uses some form of encryption, the data will be written in the clear, so if someone gains access to the files, they can see the contents. Retrofitting encryption/decryption code to existing WebSphere MQ applications is not a trivial exercise. It may need logic to handle different severities of message; it may need

to handle ASCII to EBCDIC changes and may also need some form of key management application.

The second concern is the ability for someone to sniff network traffic and extract message data. With WebSphere MQ 5.2 and earlier, you could use the Channel Exit mechanism to run some code to encrypt/decrypt the message. This code has the same issues of the application encryption discussed previously, although you can purchase exit code from a number of vendors.

WebSphere MQ 5.3 has added Secure Sockets Layer (SSL) to the channel mechanism. This means that messages can be sent securely between queue managers. You still have the issues of key management. Keys and certificates used by the SSL connections are stored in various repositories in the enterprise. The greater the number of WebSphere MQ components deployed, the greater the number of key registries/repositories needed. Certificates expire and need to be renewed. The longer the expiry date on a certificate, the greater the security exposure. For a large WebSphere MQ deployment, this may be an issue.

In summary, the data security issues are as follows.

- ▶ With all versions of WebSphere MQ, you have the concern that messages on persistent queues are written to disk in the form in which the application wrote them. If the application does not perform data encryption, there is a security exposure of data on disk in the clear.
- ▶ Adding encryption to applications is not a trivial exercise and may require some form of key management application.
- ▶ With MQSeries 5.2 and earlier, the only way to encrypt messages going across the network was to employ some encryption code through the channel exits. This may involve complex coding and a key management application.
- ▶ WebSphere MQ 5.3 supports SSL for channels, which provides for encryption of data. However, you still need to consider how keys and certificates will be managed.

These issues relate to a level of risk. The risk of an unauthorized user being able to see a message in a protected dataset or file is far less than that of someone being able to sniff packets on the network. So encryption of application data may be less of a concern than the use of SSL between queue managers.

Channel security

In addition to securing the data, you also need to be concerned with the security of the channels. When you start a channel, authorization is only required for local channels. Remote connection does not need any authorization. Therefore, if someone knows your TCP/IP address or hostname and channel name, they can

easily connect to your WebSphere MQ system. The port number that your listener listens to generally defaults to 1414. This is a security exposure.

If someone were able to use this exposure to install a “bogus” queue manager, they could:

- ▶ Tamper with existing messages to change message data, such as transaction amounts or passwords.
- ▶ Discard messages.
- ▶ Send floods of messages to other queue managers in an attempt to crash applications.

You can verify the partner connection as follows:

- ▶ Channel security exit
Set the security check program written by the user before the channel is started.
- ▶ Channel SSL support
Filter the Distinguished Name (DN) in the Certificate.

With WebSphere MQ 5.3 supporting SSL, you can authenticate the partner by digital certificate and define the MQSSLPEER attribute to filter the partner by DN in the certificate. However, this filtering function only is not enough, so we recommend the use of a security exit for stricter user authentication.

If you do not implement some level of channel connection authentication, you still have the risk of message addition, modification and deletion. Message modification is the greatest concern as it is the easiest to conceal (and could even be performed without the use of a rogue queue manager). To negate this risk with native WebSphere MQ, you need to use the channel exits to perform data encryption or data signing. This still carries the issues of key management.

In summary, the channel security issues are as follows.

- ▶ You should use some combination of SSL and security exits to ensure that only valid connections are made between queue managers. Use of SSL alone is insufficient, so security exits should be employed. This will involve the development and maintenance of code.
- ▶ If you want to implement protection against message modification, you need to implement either message encryption or signing through channel exists. This will involve the development and maintenance of code and key management issues.

As with the data security issues, this is a matter for risk assessment. If the network security is tight enough so that there is no way anyone could introduce a

rogue queue manager or tamper with messages, then you do not need to be concerned with this level of security.

Audit

The level of audit information and the format of audit information produced depend on the platform. On z/OS, WebSphere MQ writes SMF records that can be analyzed using the standard reporting tools.

On the non-z/OS platforms, there is little or no audit information provided by WebSphere MQ. The WebSphere MQ administration wrapper (support pac MS0E) can provide an audit log of users using the administration commands to configure OAM.

This is a significant concern for WebSphere MQ deployments across disparate platforms. An organization must be able to track security-related changes and violations for audit purposes.

Summary of native WebSphere MQ security

Native WebSphere MQ provides a number of mechanisms to apply security to a WebSphere MQ deployment:

- ▶ Use of an External Security Manager (ESM) on mainframe systems or Object Authority Manager (OAM) to provide for WebSphere MQ object access control.
- ▶ Use of SSL in WebSphere MQ 5.3 to encrypt messages travelling over a network and verify connections.
- ▶ Use of channel exits to provide user-written code to encrypt messages and verify connections.

If these mechanisms are used to provide security, you may still have issues relating to:

- ▶ Custom code management.
- ▶ Key/certificate management.
- ▶ Creation and management of access control rules and identities across many disparate platforms.
- ▶ Data security for persistent messages written to disk.
- ▶ No centralized view of access control, compliance with corporate policy and consistent audit reporting.

IBM Tivoli Access Manager for BI was developed to solve these issues and is discussed in the next section.

5.2 IBM Tivoli Access Manager for Business Integration

IBM Tivoli Access Manager for Business Integration (formerly Tivoli Policy Director for WebSphere MQ) is a value-add security management solution for IBM WebSphere MQ that greatly enhances its native security environment.

This upgrades WebSphere MQ's data protection to provide application level, end-to-end data protection for existing WebSphere MQ based applications without the need to modify or change them. This is critical for customers needing to implement an HIPAA compliant implementation of WebSphere MQ or for any customer using WebSphere MQ to process sensitive data.

IBM Tivoli Access Manager for BI also allows customers to consolidate the administration of put and get access control permissions on queues across their enterprise. Administration of these security policies is done remotely, using a Web-based tool that replaces the need for administrators to have visited each physical system.

IBM Tivoli Access Manager for BI also supports the extended WebSphere MQ family, including both WebSphere MQ Integrator and WebSphere MQ Workflow.

In the following sections, we detail how IBM Tivoli Access Manager for BI addresses the security and management issues raised in the previous section.

5.2.1 Solving the issues with IBM Tivoli Access Manager for BI

In this section, we deal with the four key areas of concern raised in the last section:

- ▶ Access control
How does IBM Tivoli Access Manager for BI control who has access to the queue managers, queues and other WebSphere MQ objects?
- ▶ Data and channel security
How does IBM Tivoli Access Manager for BI stop people seeing and modifying message contents?
- ▶ Audit
How does IBM Tivoli Access Manager for BI track security related events across an WebSphere MQ implementation?

These issues are discussed in the following sections.

Access control

IBM Tivoli Access Manager for BI removes the need for the OAM function to control access of PUT/GET operations against queues, that is, the normal day to day application access of queues. OAM is still required for access control to other WebSphere MQ objects and actions, such as administrative actions. It introduces an interceptor layer between the application API calls and the queue manager. This is shown in Figure 5-2.

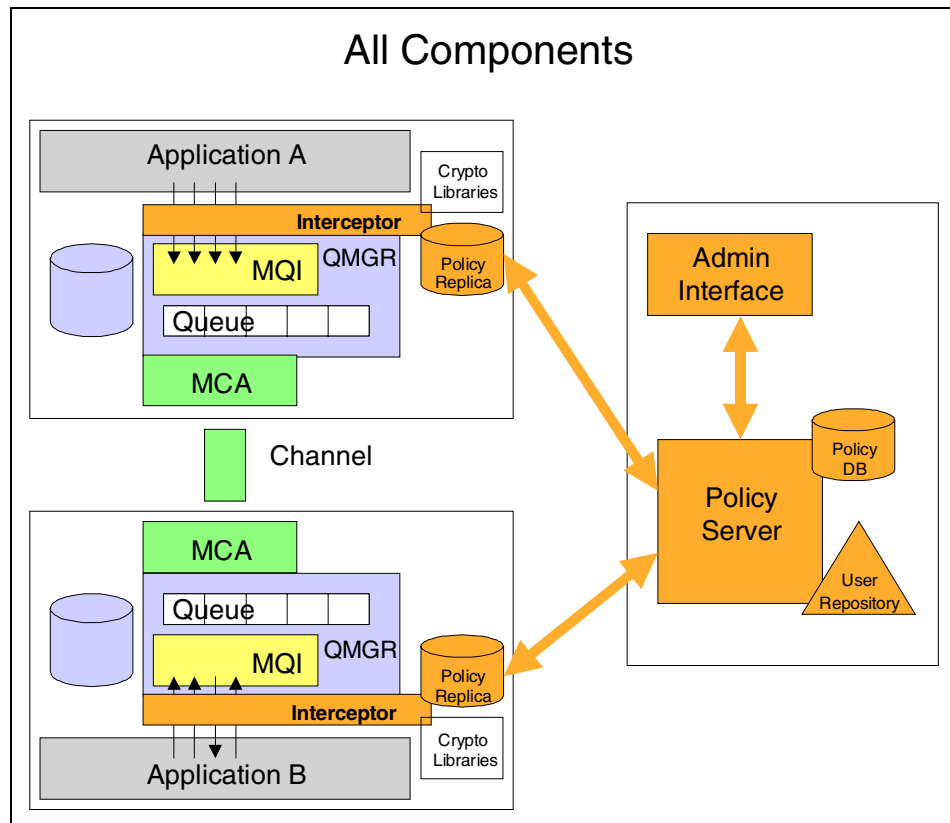


Figure 5-2 WebSphere MQ data flow with IBM Tivoli Access Manager for BI interception

The interceptor's job is to act as a security layer over WebSphere MQ. It captures all requests made to WebSphere MQ for service and specifically focuses on enforcing access control on queue open requests and data protection on put and get requests to queues. The interceptor process does not replace the standard MQI library but instead sits over it.

Authorized service requests are passed on to WebSphere MQ for processing. Unauthorized requests are rejected with a standard WebSphere MQ return code.

The interceptor fully replicates the standard WebSphere MQ API library and supports server applications using the MQI, JMS and AMI programming APIs.

Note: IBM Tivoli Access Manager for BI's support for JMS is available only when using the "bind" transport, that is, the WebSphere MQ server must be installed on the same machine as the JMS application. "Client" transport is not supported.

Authorization decisions are made by the interceptor using security policy data set by the customer at the central security policy manager (Access Manager) and replicated out to the interceptor. This replication is done over a SSL secured TCP/IP connection, ensuring the security of the policy data as it traverses the network. The local policy replica data store is a complete copy of all of the authorization and data protection policy information applicable to the queues on that system. This replica is copied from the master copy at the central policy manager when the IBM Tivoli Access Manager for Business Integration server (daemon) is started.

Having a complete local copy of security policies reduces network overhead, allows for higher message throughput and provides a highly available environment since message processing can continue even if the central security policy manager is down.

The local replica of the security policy data is backed up to disk to ensure availability upon system restart. When policy changes are made to the master security policy database, an updated copy of the data is replicated out to each system where the interceptor is installed.

A second authorization check is also made on the target system of a message when an application there issues a get request. This is primarily intended to identify messages that are injected into the network from an unauthorized entry point.

We have not discussed how IBM Tivoli Access Manager for BI performs its authentication. A part of the central security policy manager component (Figure 5-4 on page 113) is an LDAP directory that is used to store and validate the credentials of applications requesting services from WebSphere MQ.

When an application first opens a queue, the interceptor process queries the local operating system to determine the operating system ID (OS ID) under which the requesting application is running. This OS ID is then mapped to a certificate using a configuration file that is securely stored on that system. The distinguished name (DN) field in the certificate is mapped to an Access Manager user and authorization checks for WebSphere MQ access are performed using this Access Manager user. If the credential is valid, it is then used by the IBM

Tivoli Access Manager for BI server (daemon) to investigate the local security policy replica to determine whether the application has the proper permission to open that queue for either put permission, get permission, both or none. Once this credential is validated, it is cached and reused for as long as the application makes active use of the connection or until the connection is closed. This reduces network traffic and increases average message throughput.

For a highly available environment, it is recommended that this directory be replicated to ensure that processing is not interrupted. IBM Tivoli Access Manager for BI bundles a licensed copy of the IBM Directory Server for IBM AIX, Sun Solaris and Microsoft Windows 2000 and Windows NT. The IBM LDAP Directory is also bundled with IBM OS/390® and IBM z/OS. Customers can configure IBM Tivoli Access Manager for BI to use the directory on any of these platforms. If a customer has already deployed the Netscape/iPlanet LDAP Directory, it can be used instead.

As mentioned in the previous section, identity management may be an issue for larger deployments, where you have accounts on various systems and access control now defined in IBM Tivoli Access Manager for BI. IBM Tivoli Identity Manager (ITIM) works closely with the Access Manager family and can manage all users' accounts from a single console.

Figure 5-2 on page 108 also shows the Access Manager common components. The central security policy manager component is shown as two processes and two data stores. The policy manager process and its accompanying data store, the master security policy database and the LDAP directory have been described above.

The other process is the administration tool. This is a Web-based application that provides an easy to use GUI for setting, viewing, and updating security policies.

Protected queues and queue managers are represented in the resource namespace as objects, and may be secured by setting a policy on the desired queue or queue manager. A companion utility is also provided to automate the population of this resource space with the existing queue manager and queue definitions on each system to be managed.

The administration tool also supports the ability to delegate responsibility for just a subset of the defined queue managers or queues to a junior administrator. IT organizations may find this feature useful since it allows them to maintain control over the total enterprise security infrastructure for WebSphere MQ but still grant a specific department or line of-business the ability to manage the subset of resources they use or own.

All administration can also be done through scripting using a command line interface that is also available.

Data and channel security

IBM Tivoli Access Manager for BI provides the means to sign and encrypt message data before it is written to disk and over the network. Figure 5-3 shows the cryptography libraries used to achieve this.

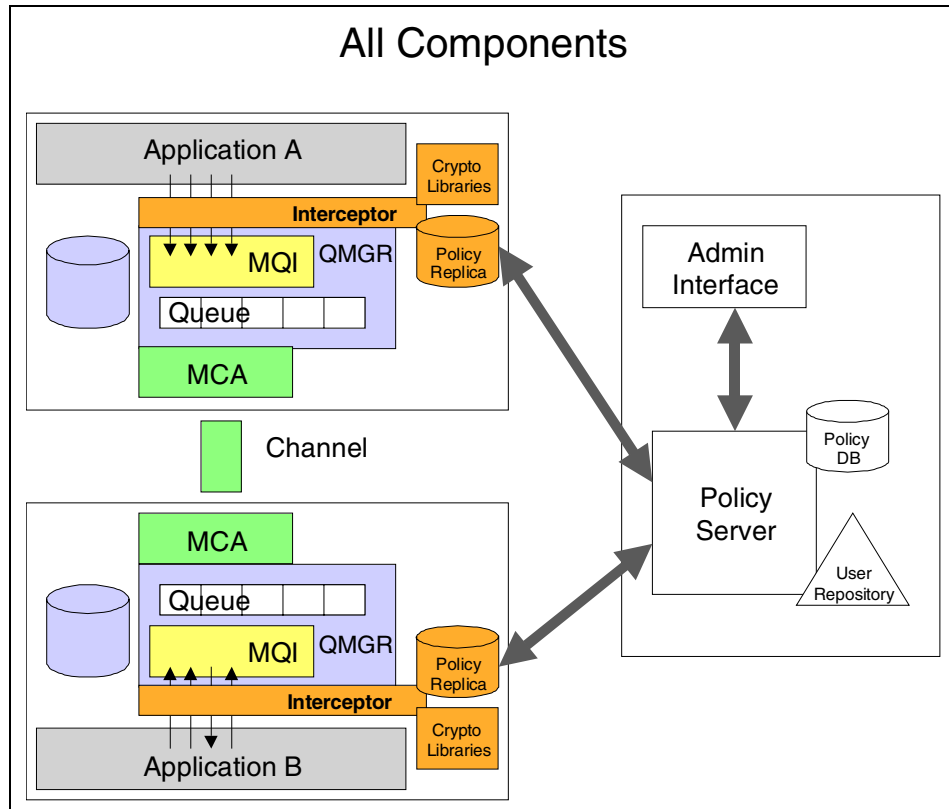


Figure 5-3 WebSphere MQ data flow with IBM Tivoli Access Manager for BI interception and cryptography

The cryptography libraries are used by the interceptor to enforce the data protection policy set on each queue. They provide the asymmetric cryptography services used to sign the data portion of a message as part of message put processing and then verify that signature as part of message get processing. This allows the verification of whether or not a message has been changed while being processed by WebSphere MQ or during transport from system to system on the network. Also provided are the symmetric cryptography services used to encrypt the data portion of a message as part of message put processing and then decrypt it as part of message get processing.

The data protection policies options that IBM Tivoli Access Manager for BI supports are:

- ▶ NONE
No data protection, do not sign or encrypt the message data
- ▶ INTEGRITY
Sign message data to allow the verification of integrity
- ▶ PRIVACY
Sign and encrypt message data for integrity and confidentiality

IBM Tivoli Access Manager for BI uses public and private keys to perform its data protection functions. Key pairs can be associated with specific applications or shared by all applications on a server. IBM Tivoli Access Manager for BI can utilize PKI credentials generated by a variety of third party Certificate Authorities including VeriSign, Entrust, Baltimore and Netscape in addition to the self-signed certificates it can generate itself. A software-based key ring is also provided for the secure storage of these credentials on each WebSphere MQ server.

Audit

An Audit posture is also part of each policy. If auditing is enabled, customers are provided XML style formatted records that document the success or failure of attempts to open and close queues and put and get messages. The specific audit options are:

- ▶ ALL
Records all auditable events.
- ▶ NONE
Auditing function is turned off.
- ▶ PERMIT
Records only successful accesses.
- ▶ DENY
Records only denied requests for access.
- ▶ ADMIN
Records OPEN, CLOSE, PUT, and GET operations on protected WebSphere MQ queues.
- ▶ ERROR
During an MQGET, when an error occurs and IBM Tivoli Access Manager for BI puts the message on the error queue, if the ERROR audit option is set, an audit record is generated.

These XML files can be centralized (perhaps using WebSphere MQ) and merged to provide consolidated reporting.

How does it work?

How does a transaction work with IBM Tivoli Access Manager for BI and WebSphere MQ?

To summarize how IBM Tivoli Access Manager for BI secures a WebSphere MQ transaction, let's review the example shown in Figure 5-4.

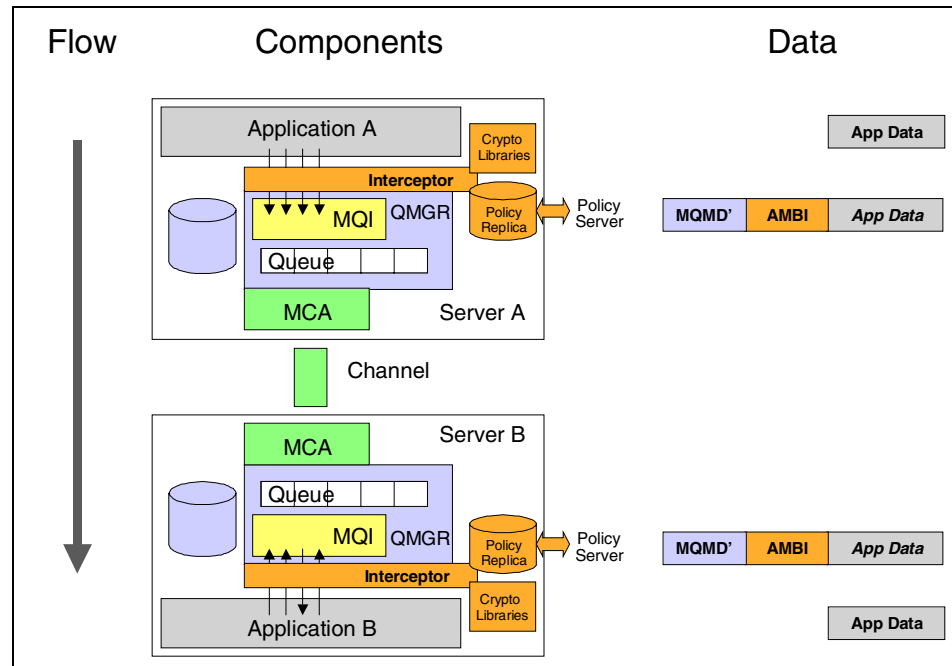


Figure 5-4 WebSphere MQ transaction with IBM Tivoli Access Manager for BI

This figure illustrates a single message in its path from a source application (Application A) to a target application (Application B) and shows the transformation made to secure the message and carry along additional point-of-origin information.

When the application on Server A puts a message, the format of the resulting message would be a WebSphere MQ specific MQMD header followed by the message data. The data is in a clear text format that may be easily interpreted by any hacker, disgruntled employee or a vendor; the premise is that they can get their hands on it.

Since IBM Tivoli Access Manager for BI is installed on this system, the message is captured by the interceptor, which on most platforms runs within the memory space of the WebSphere MQ application itself.

In this example, the data protection policy, set on the queue the message is directed to, is “PRIVACY”, which means that messages put to that queue will be encrypted and then signed.

As the message passes through the IBM Tivoli Access Manager for BI security layer, the data portion of the message is encrypted and signed following the industry standard PKCS#7 data format. This standard specifies that the symmetric encryption key generated to encrypt the message will itself be encrypted using the public key assigned to the target application. This allows the encryption key to be passed along with the data in a secure format, avoiding the need to do some form of “out of band” key exchange. It also ensures that no one can decrypt the message without the private key of the target application.

This encrypted data is then hashed and the hash is signed using the private key assigned to the sending application. The encrypted data, the encrypted encryption key and the signed hash, or digital signature form the “App Data” portion of the new message shown in Figure 5-4 on page 113.

As mentioned previously, a second authorization check will be done on this message when the application on Server B issues a get to read the message out of the target queue. The credential used for this check is not the OS ID from the application on Server A that is stored in the MQMD header of the message. Server B may be a completely different platform from Server A running a different operating system where the OS ID would have no meaning. To address this issue, when constructing the new protected version of the message, the interceptor process will append an additional security header to the now protected data to carry a more globally identifiable credential. This security header is illustrated in Figure 5-4 on page 113 as the “Access Manager for Business Integration” box fronting the protected data. The additional credential used is the public key certificate assigned to the sending application on Server A.

In addition, the security header can carry any request made by the sending application for WebSphere MQ to perform some form of data conversion on the message data. When using IBM Tivoli Access Manager for BI, the message data may already be in a protected format, by the time it gets to WebSphere MQ, preventing WebSphere MQ from carrying out any conversion requests.

To maintain application compatibility, the interceptor also reviews the message MQMD header to look for a data conversion request. If it finds one it will mark the security header to indicate a data conversion request and then reset the field in the MQMD to `No conversion requested` to prevent the queue manager process from trying to do a conversion on the protected data. At this point, the new

message (MQMD header minus the data conversion request [security header made up of Server A application's certificate plus data conversion request and the protected data]) is passed onto WebSphere MQ using an **MQPUT** command which the interceptor process issues. The entire data payload of the newly protected message is formatted to follow the PKCS#7 standard as described above.

WebSphere MQ then processes the new protected message as it normally would, minus the data conversion. When the application on Server B issues a get request, it is passed through the interceptor and on to WebSphere MQ.

WebSphere MQ returns the message to the interceptor. The interceptor then goes through the following steps before passing the message back to the requesting application:

1. First, it will check the data protection policy on the queue and verify that the message matches it. As an example, if the policy is **PRIVACY** and the message is not signed and encrypted, it will be directed to a special error queue. An audit event will also be generated if the security policy specifies to generate one on this condition.
2. Next, it will verify the signature on the message. If this check fails, the message will again be directed to a special error queue and an appropriate audit event generated if requested.
3. Next, it will decrypt the message data by decrypting the symmetric encryption key with the private key of the application requesting the message and then using it to decrypt the message data. Again, any failure results in the same actions as described above.
4. Next, it will check the distinguished name field in the certificate contained in the security header to see if the application that generated the original message is authorized to have put the message into the queue where it appeared. Again, any failure results in the same actions as described above.
5. The last step in message get processing by the interceptor is to perform any data conversion required as indicated in the security header. This is done by passing the clear text data to the same conversion routines in WebSphere MQ that the queue manager process would have called if it had done this processing.

The reconstructed message is now returned to the requesting application on Server B. Remember that the interceptor process performing this validation and restoration of the message to its original form is typically running in the memory space of the application on Server B. As a result, the message has had true application level integrity and confidentiality enforced on it with demonstrable audit records, if requested.

Summary of what IBM Tivoli Access Manager for BI provides

A security management solution for WebSphere MQ must recognize and address an enterprise's messaging security requirements. It should provide a single point of administration for security policy, application level, message integrity and confidentiality, high performance and scalability. IBM Tivoli Access Manager for BI was developed specifically to address these issues. It is the first enterprise level, security management solution for WebSphere MQ. It provides centralized administration of data protection and queue, put and get, permission policies. It is a highly scalable, security solution that can transparently secure WebSphere MQ applications without modification. It also can generate detailed auditing records showing that transactions were expressly authorized and properly protected.

IBM Tivoli Access Manager for BI supports IBM WebSphere MQ Integrator, allowing customers to secure messages to and from a WebSphere MQ Integrator hub. It also supports IBM WebSphere MQ Workflow, allowing customers to securely exchange workflow documents from system to system as they are being processed.

5.3 A technical look at IBM Tivoli Access Manager for BI

In this section, we take a look at how IBM Tivoli Access Manager for Business Integration is put together and operates. We also look at what the current product (IBM Tivoli Access Manager for Business Integration V4.1) supports and how it may evolve in the future.

5.3.1 Access manager and IBM Tivoli Access Manager for BI

IBM Tivoli Access Manager for BI is a member of a family of access management solutions from IBM that share a common set of services and have shared functionality. This family is known as the Access Manager family.

Access manager architecture and decision model

Figure 5-5 shows the authorization decision model of Access Manager.

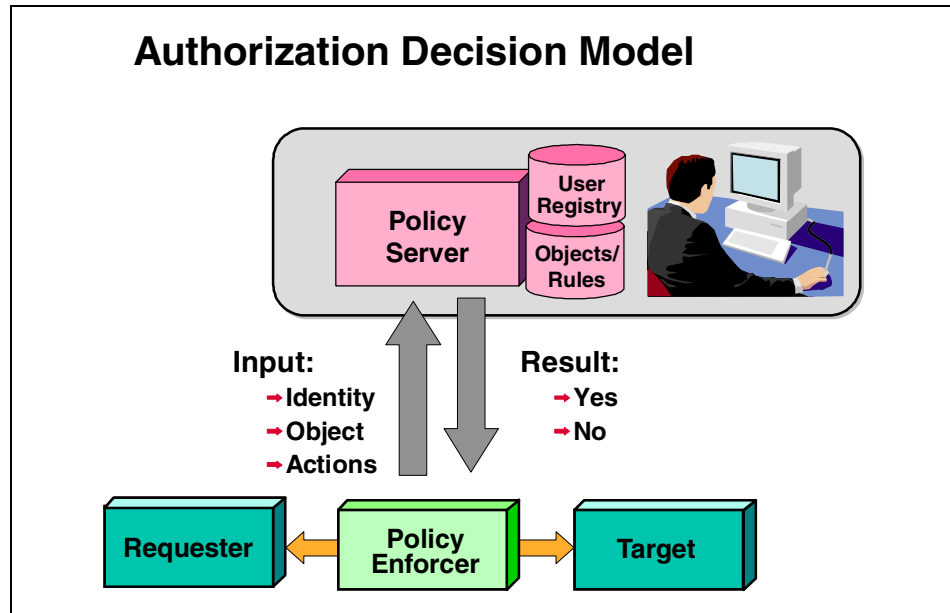


Figure 5-5 Access Manager authorization decision model

This is a straightforward model which is centralized. The unified security policy is clearly established, and once established, it becomes the rules that the policy enforcers use to determine which requesters can access which targets and in what ways.

The answer to each authorization request is yes or no.

The components in this model are:

- ▶ The User Registry, which contains all users and groups that are defined to the installed Access Manager products (including IBM Tivoli Access Manager for BI). It is an LDAP compliant directory, such as the IBM Directory Server.
- ▶ The objects/rules database is the policy database. It contains all resources that are used by the policy enforcers (such as queues and queue managers), ACLs applied to those resources, and policies applied to the resource (such as the privacy attribute on queues).
- ▶ The Policy Server is the Access Manager Policy Server. It maintains the master (read/write) copy of the policy database and controls the replication of this database with the local copies. It communicates with the policy enforcers via the Access Manager runtime libraries (called pdrte).

- ▶ The Policy Enforcer is the Access Manager product that will enforce the policy, such as IBM Tivoli Access Manager for BI. The local Access Manager runtime component (AMRTE or pdrte) will perform the authorization checks, with questions such as “Can this identity run this action against this object?” and “Can this user put data into this queue?” It uses the local (read-only) copy of the policy database for this check.
- ▶ The requester is the user or application that needs to access the target (such as a queue).

Figure 5-6 shows a number of the policy enforcers.

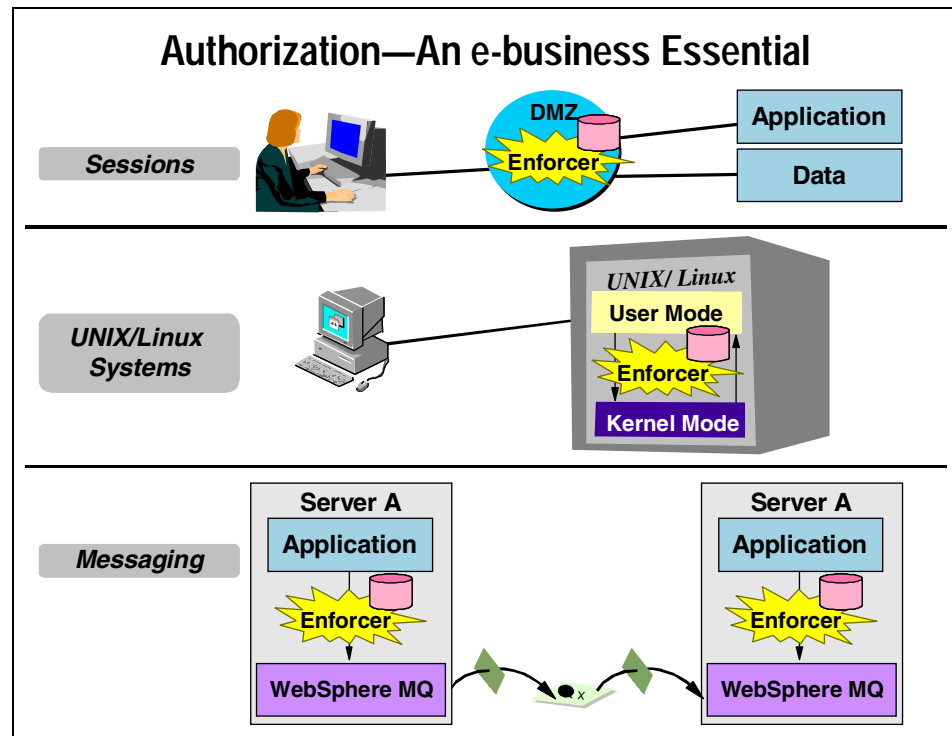


Figure 5-6 Access Manager policy enforcers

The policy enforcers shown here are:

- ▶ Sessions
This is the Access Manager for e-Business (AMeB) product that provides security to Web applications via a reverse proxy called WebSEAL.

- ▶ UNIX/Linux Systems

This is the Access Manager for Operating Systems (AMOS) product that provides enhanced security to UNIX/Linux OS resources, such as files and network connections.

- ▶ Messaging

This is the IBM Tivoli Access Manager for BI product.

There are also Access Manager products that provide security to J2EE applications (hosted by WebSphere Application Server or BEA WebLogic), portal applications (through WebSphere Portal Server or Plumtree) and other third-party applications.

All products utilize and ship the same set of shared services described above. A single instance of these shared services can support the installation of all three products, allowing a customer to consolidate the administration and management of security policy across WebSphere MQ queues, Web resources, UNIX/Linux resources and application resources.

Access Manager administrative concepts

All of these products are part of the same Access Manager family and can use the same Access Manager infrastructure. This means all users and groups can be defined in a single user repository for accessing multiple applications and systems. Also, all access control can be defined in a single place and have a single administrative interface (the Access Manager Web Portal Manager). Figure 5-7 on page 120 shows the administrative concepts for all Access Manager products.

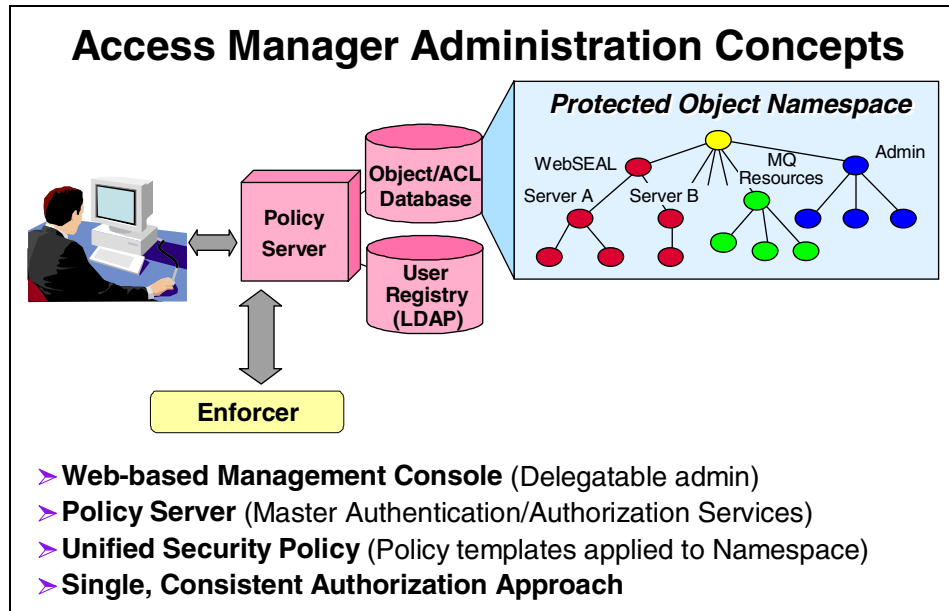


Figure 5-7 Access Manager administrative concepts

The key concepts in Access Manager administration are:

- ▶ **Web-based management console**

There is a single Web-based management console. This means that administration client code does not need to be deployed throughout an enterprise. A delegated administration model can be deployed for all Access Manager products, where administrators can only administer objects that they have been granted admin rights to.
- ▶ **Policy server**

Access Manager uses a central master authentication and authorization service, so all administration is performed against a single source of information. This information is then replicated to the enforcers so that the enforcers can operate in a disconnected mode.
- ▶ **Unified security policy**

All resources and ACLs are mapped to a single, unified namespace. Policy templates can be applied to all objects in the namespace. The namespace is hierarchical, allowing for inheritance of ACLs and policy to minimize the definitions to be deployed.

- ▶ Single, consistent authorization approach
All authorization is performed against a single user registry (although there may be replicas deployed for performance reasons). Thus, all users and groups are centrally managed for all systems and applications that the Access Manager products are securing. Thus, a wide ranging set of users are managed consistently through a single interface.

In the following sections, we deal with IBM Tivoli Access Manager for BI specifically.

5.3.2 IBM Tivoli Access Manager for BI Architecture

The components of IBM Tivoli Access Manager for BI are shown in Figure 5-8.

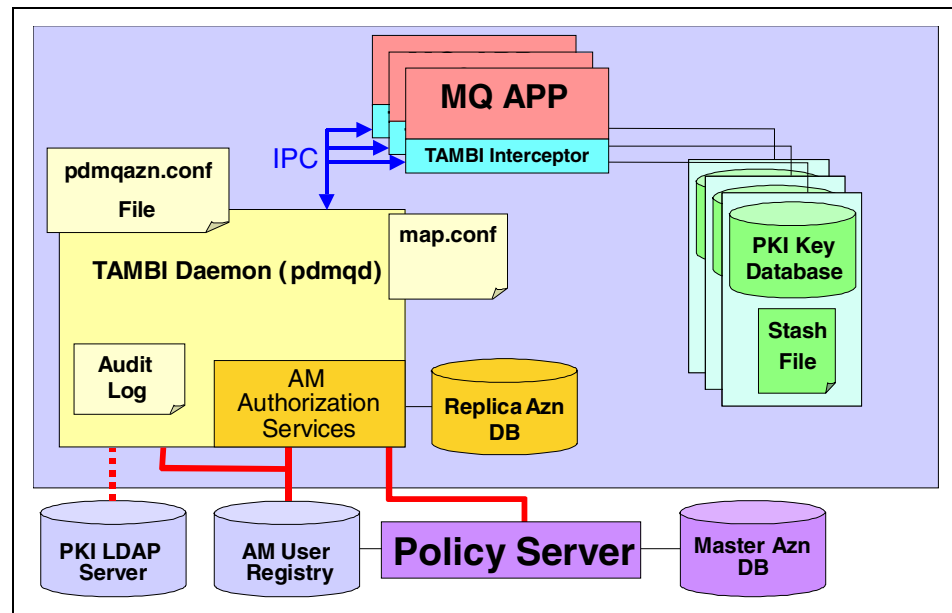


Figure 5-8 IBM Tivoli Access Manager for BI components

The components are:

- ▶ The IBM Tivoli Access Manager for BI server (daemon), with AM runtime, replica of the policy database and some configuration files.
- ▶ The IBM Tivoli Access Manager for BI interceptor and key databases.
- ▶ The Access Manager components, including the Policy Server, Master Policy database and User Registry.

The Access Manager components have been discussed in the previous section. The IBM Tivoli Access Manager for BI interceptor and server (daemon) components are discussed in the next sections.

IBM Tivoli Access Manager for BI interceptor

Figure 5-9 shows the IBM Tivoli Access Manager for BI interceptor components and how they interact with the applications and local queue manager.

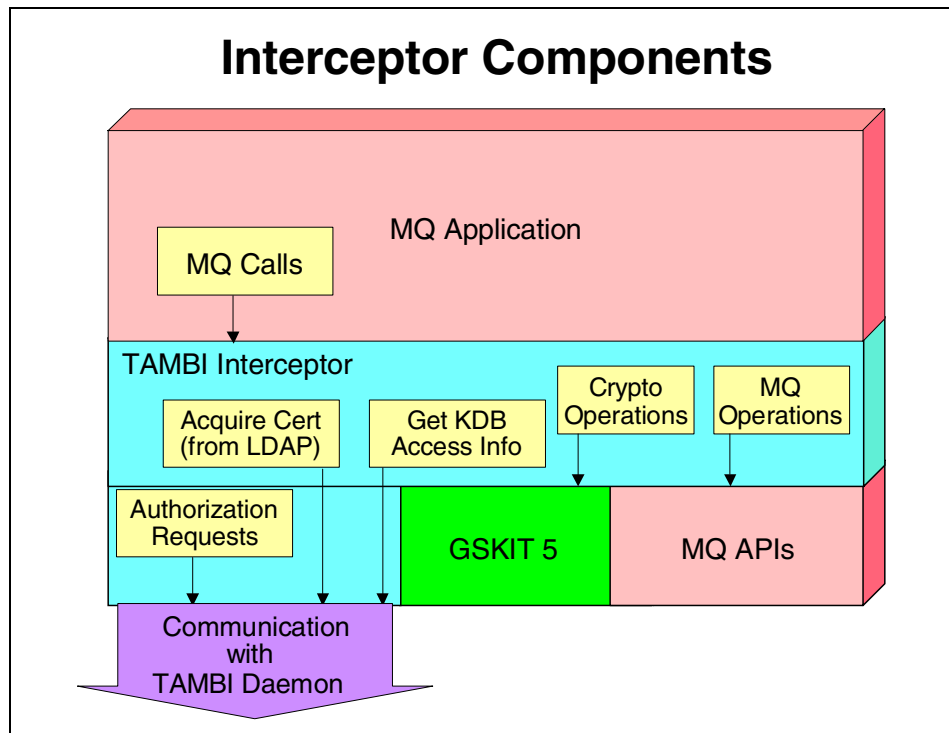


Figure 5-9 IBM Tivoli Access Manager for BI interceptor

The figure shows how a IBM Tivoli Access Manager for BI interceptor running as part of an WebSphere MQ Application uses the IBM Tivoli Access Manager for BI server (daemon) via IPC, IBM Global Security Toolkit (GSKIT) and WebSphere MQ to service WebSphere MQ calls from the WebSphere MQ application.

When an WebSphere MQ application on a machine protected with IBM Tivoli Access Manager for BI connects to a queue manager, the IBM Tivoli Access Manager for BI interceptor is initialized. It runs as part of the WebSphere MQ application.

When the WebSphere MQ application makes WebSphere MQ calls, these are received by the IBM Tivoli Access Manager for BI interceptor rather than by the WebSphere MQ code. After the request is processed, it can be passed onto the WebSphere MQ code (possibility after modification) or rejected.

In order to decide whether a given request should be allowed, the IBM Tivoli Access Manager for BI interceptor calls the IBM Tivoli Access Manager for BI server (daemon) over IPC. It also uses the daemon to acquire PKI information for accessing KDB files and to acquire recipient certificates that are stored in LDAP. It also uses the IBM Tivoli Access Manager for BI server (daemon) to generate audit records.

The interceptor uses the GSKIT to provide message signing and encryption functions. In order to determine which KDB file to open for the application, the interceptor passes the OS identity of the application to the daemon which then passes back the file name of the KDB file containing the PKI identity for the OS identity.

IBM Tivoli Access Manager for BI server (daemon) functions

Figure 5-10 on page 124 shows the IBM Tivoli Access Manager for BI server (daemon) components. Note that the figure does not show the Audit components.

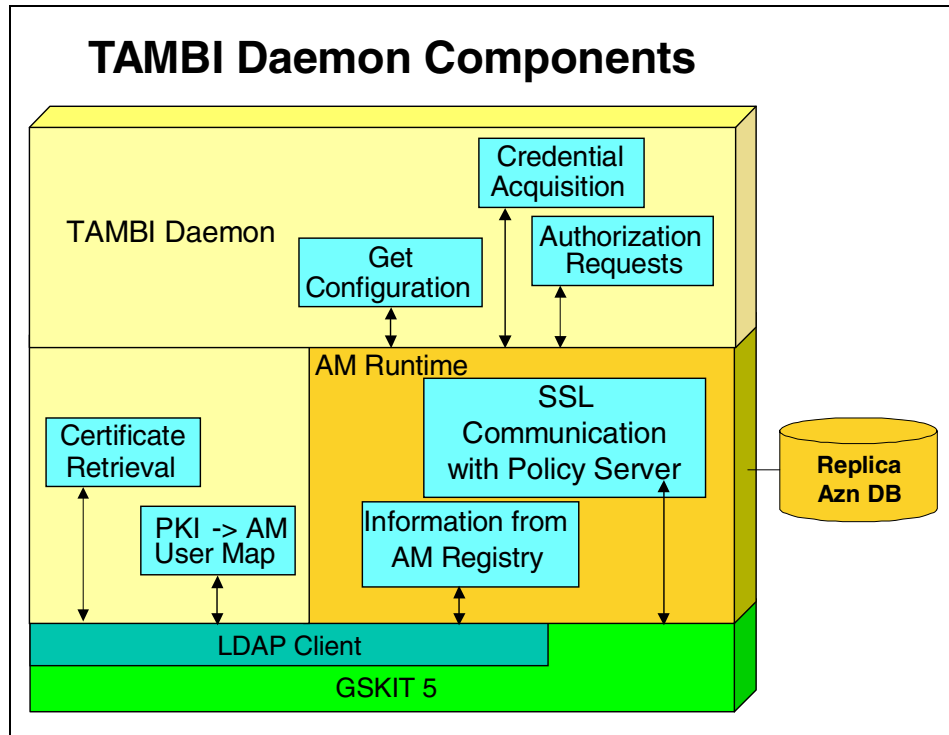


Figure 5-10 IBM Tivoli Access Manager for BI server (daemon)

The IBM Tivoli Access Manager for BI server (daemon) is a local mode aznAPI application. It calls Access Manager Authorization Services to get access decisions. The Access Manager Authorization Services communicate with the Access Manager Policy Server and the AM User Registry.

Configuration for the Access Manager Authorization Service is provided by the pdmqazn.conf configuration file. This contains the aznAPI configuration parameters required to communicate with the Policy Server.

The IBM Tivoli Access Manager for BI server (daemon) is responsible for user identity mapping. It uses the map.conf file (or interactive login on Windows) to associate an application attempting to access WebSphere MQ with a PKI identity.

The server (daemon) is also responsible for mapping a PKI identity (either the one associated with a local application or one received in a message) to an Access Manager user. This is done using the secPKIMap information in the Access Manager User Registry.

If certificate retrieval from LDAP is being used, the server (daemon) can also contact an LDAP server that contains the certificates used for message encryption.

The final responsibility of the IBM Tivoli Access Manager for BI server (daemon) is auditing. It creates an audit file and records access to AMB resources based on the auditing options specified (by policy applied to the objects).

5.3.3 IBM Tivoli Access Manager for BI V4.1

In this section, we look at the WebSphere MQ support provided by the current version of IBM Tivoli Access Manager for BI, version 4.1. The following is a summary of what is supported and not supported with IBM Tivoli Access Manager for BI V4.1.

IBM Tivoli Access Manager for BI V4.1 supports the following products in the WebSphere MQ suite:

- ▶ WebSphere MQ 5.3
- ▶ IBM MQSeries 5.2 with CSD's 3 or 4 or 5
- ▶ WebSphere MQ Integrator V2.1
- ▶ WebSphere MQ Integrator Broker V2.1
- ▶ MQSeries Integrator V2.02
- ▶ MQSeries Workflow V3.3

It also supports the use of IBM 4758 and 4960 hardware crypto cards.

The platforms that IBM Tivoli Access Manager for BI V4.1 will run on are:

- ▶ AIX 4.3.3 and 5.1
- ▶ Solaris 7 and 8
- ▶ Windows NT and Win2K
- ▶ OS/390 R10 and z/OS R1 and higher

The following PKI applications/vendors are supported with IBM Tivoli Access Manager for BI V4.1:

- ▶ Tivoli
- ▶ VeriSign
- ▶ Entrust
- ▶ Netscape
- ▶ Baltimore

Certificates from other X.509 V3 compliant CAs are likely to be compatible with IBM Tivoli Access Manager for Business Integration V 4.1.

The current version has the following limitations.

- ▶ Support for server only
There is no support for WebSphere MQ clients today, either MQI client or JMS pure Java client
- ▶ Supports all MQI features except:
 - Application Message Segmentation
 - Channel data conversion
 - Use of non-threadsafe MQI libraries
- ▶ Support for TCP/IP transport only
This is a test statement only

So if you have a large deployment including WebSphere MQ clients or other systems that are not currently supported (AS/400, HP/UX or Linux), then IBM Tivoli Access Manager for BI V4.1 may not be appropriate for a deployment.

5.3.4 IBM Tivoli Access Manager for BI future evolution

As highlighted in the previous section, there are a number of areas that the current version of IBM Tivoli Access Manager for BI does not address.

The following are the stated directions for IBM Tivoli Access Manager for BI product development:

- ▶ Additional platform coverage (HP/UX, Linux Intel and WebSphere MQ clients)
- ▶ Support for WebSphere MQ V5.3 and DCE free Access Manager runtime on z/OS
- ▶ Additional host application coverage (IMS, CICS Bridge and IMS Bridge based apps)
- ▶ Uniform product function across distributed and mainframe versions
- ▶ Data protection API for WebSphere MQ applications (support for custom WebSphere MQ applications)
- ▶ Support for the new WebSphere MQ API Interface to enable faster platform support
- ▶ Integration with balance of WebSphere BI Suite; formalization of IBM ICS support, and possible bundles with WebSphere BI products

This is only a stated direction and not a commitment for future product functionality.

5.4 Summary

In closing this chapter, we revisit the security functionality provided by WebSphere and the functionality provided by IBM Tivoli Access Manager for BI and then discuss when you would use IBM Tivoli Access Manager for Business Integration.

5.4.1 Native WebSphere MQ Security vs. IBM Tivoli Access Manager for BI Security

In summary, native WebSphere MQ security provides:

- ▶ Authentication based on OS ID of an application (but may be tied to individual users)
- ▶ Authorization administration which must be performed at each server (but may be simplified by using some WebSphere MQ tools)
- ▶ Data which can be secured in transport using channel exit coding or SSL transport, but not while in a queue (unless application-level encryption is employed)

When using IBM Tivoli Access Manager for BI:

- ▶ Authentication is based on certificates associated with each application
- ▶ Message data is protected end-to-end without application change
- ▶ Security policy is centrally administered across all supported servers
- ▶ Much finer granularity in security policies can be implemented

5.4.2 When to use IBM Tivoli Access Manager for BI

So when would you use IBM Tivoli Access Manager for BI over native WebSphere MQ security? Each of the following scenarios would benefit from IBM Tivoli Access Manager for BI.

- ▶ Where there are large number of distributed systems running WebSphere MQ.

With native WebSphere MQ security, you would have to manually configure OAM on each of these systems and SSL connections between each queue manager on each system. With IBM Tivoli Access Manager for BI, you still

have to install the product on each system, but from there on you can centrally manage all policy configuration and data encryption.

- ▶ Where security policy compliance and audit reporting is a key focus of the organization.

With native WebSphere MQ security, you would need to put procedures in place to ensure that the manual configuration of OAM complies with corporate policy. From an audit perspective, not all platforms support logging audit information for WebSphere MQ object access, and if they do, the format is different on each system. Where OAM components write audit logs, it is an “all or nothing” implementation. You cannot restrict audit logs to individual objects. With IBM Tivoli Access Manager for BI, all administration is performed centrally and policy can be enforced. Each IBM Tivoli Access Manager for BI installation will write audit logs in a standard XML format and logging is based on the policy applied to each object.

- ▶ Where much of the WebSphere MQ access is tied to individual users rather than generic application logins.

Where individual users are associated with access to queues, rather than application logins, there is a huge overhead of identity management. When a user is added to a system, removed from a system or if their role changes, the OAM definitions need to be updated to reflect the changes. With native WebSphere MQ security, this is a large administrative overhead. IBM Tivoli Access Manager for BI, in concert with IBM Tivoli Identity Manager, can significantly reduce this overhead and provide centralized administration of all identities and their access.

The current version of IBM Tivoli Access Manager for BI does not support all platforms, nor does it manage WebSphere MQ clients. However, these platforms and the clients are firmly in the sights of development and should be available in pending releases.

So for small, simple WebSphere MQ deployments, you may not need IBM Tivoli Access Manager for BI. You could manage all security definitions, policy compliance, data security and key management and audit requirements with a manageable amount of customization. However, for larger, more complex, multi-platform deployments (the true focus of WebSphere MQ), IBM Tivoli Access Manager for BI would be a better security solution than trying to deploy, configure and manage all of the native WebSphere MQ security components.



Part 2

Securing WebSphere MQ

We have separated this redbook into two distinct parts, a theoretical part and a practical part.

This is the second part, in which we implement technology and document exactly what we needed to do in our case scenario and how we did it in a manner that can be replicated by you, in your own environment.

As in Part 1, we have divided this part of the book into five distinct chapters.

Chapter 6, “Management issues” on page 131 lays out a business scenario. We have created a fictional bank for our project, which we have named Dolphin Bank. We chose a bank as a business scenario because we are writing about security and security has a monetary component.

Hopefully, the techniques outlined in this part of the book are applicable to other vertical industries, because if you can protect money, you stand a fair chance of protecting more mundane business objects.

Chapter 7, “Business scenario” on page 137 lays out the hardware and software architecture that our bank has acquired and built over the years.

Chapter 8, “Business scenario architecture” on page 141 shows our “before” configuration. Like most enterprises, our bank rates security as a high priority issue when talking about it, but is considerably less keen when it involves spending precious resources on the subject. We therefore document the “default” security that most customers inherit when they install WebSphere MQ Series. We capture screens, scripts and configuration objects for your edification.

Chapter 9, “Business scenario security configuration” on page 167 looks at the vulnerabilities that exist following implementation of the base product set. It discusses where new technologies such as SSL can be effectively deployed to mitigate risk, and it also looks at other techniques that an enterprise may wish to deploy to “fill the holes” that SSL does not.

In Chapter 10, “Architectural vulnerabilities” on page 177, we conclude this part of the book with our solution. We implement WebSphere MQ security to the best of our ability in a complex environment. We are cognizant that most enterprises will be retrofitting SSL security and OAM enhancements into an existing production environment; this is what we have emulated and documented.

The redbook itself does not end with Chapter 10. We thought it important to add a section on hints and tips, a section on further information sources and a section that contained program source, JCL, and scripts suitable for reviewing. Finally, the appendixes at the end of the redbook provide readable, useful addenda to the information already provided.



Management issues

An enterprise without a well thought out security policy is doing a disservice to its employees and to its stockholders. Because middleware is ubiquitous and largely unseen, it tends to be ignored when it comes to security planning. Many organizations, in fact, will spend more money on security guards patrolling an office than on developing and implementing an infrastructure security policy.

The nature of security exposures is such that they are unwelcome and that they arrive unexpectedly. People do not handle unexpected events well and do not like thinking about them. But just as disaster planning and recovery have re-emerged as a business imperative in recent times, all it will take is one major middleware security breach to panic world markets and to make management absolutely serious about middleware security. We would rather have our customers ahead of such an event rather than behind it.

6.1 System validation

This section discusses problem definition and attack simulations.

6.1.1 Problem definition

Having developed a policy and implemented it using technical solutions, how do you ensure that the system is capable of being protected from the risks and threats you determined?

The problem with security is that we cannot define exactly what it is to do. While we believe we have identified all the threats with some form of attack tree analysis, we cannot protect from attacks that have not been identified.

Here is a simple example: two queue managers communicate using send and receive channels. SSL is enabled for the channels. Messages passing through the channels are enciphered and signed as required by the policy documentation.

Using a TCP/IP network packet sniffer to specify the correct listener port and IP address, we harvest a packet from the network and note that it is unreadable.

An auditor might then place a checkmark on a review document stating that the message was unreadable, showing conformity with the corporate standards. The test has been passed. However, this is only half the story. What consideration have we given to the key strength? How robust is the cipher we have used? With whose key has the message been enciphered? What exploits still exist that would allow an attacker to convert the ciphertext to plain text?

The point of this example is to show something that we already know. Enterprise security is composed of many pieces, all of which need to be examined at a much higher level within an enterprise than single technology viewpoints; they all need to be examined in concert as well.

6.1.2 Attack simulations

Since most enterprises conduct regular Disaster Recovery (DR) simulations, one of the most effective ways of validating security is to simulate an actual attack.

From a management standpoint, DR drills are expensive to implement and are logistical challenges. Security attack simulations are no different, and should perhaps be considered to be part of such efforts, being financed under the umbrella of DR. The logic is that any significant security “event” would likely constitute a disaster for the enterprise, resulting in invocation of the DR policies anyway.

From a management standpoint, having existing staff involved in attack simulations is generally not considered a good idea.

Staff members are not usually encouraged to attack their own systems for a number of reasons:

- ▶ There is fear of hurting co-workers through exposure of known security flaws.
- ▶ Line managements are of the opinion that any flaws discovered will reflect badly on them.
- ▶ Line managements do not want to encourage insiders to attack systems in case it gives them “ideas” best left dormant.
- ▶ There is the simple fact that attacking systems is considered “fun” by most of us in the IT industry, and management usually has a hard time allowing this.

Because of these and similar reasons, organizations have started to emerge that provide attack services for a fee. There are several advantages associated with outsourcing security attack simulations to such organizations.

- ▶ Those responsible for conducting the attacks learn from each engagement they perform and this cumulative experience base is usually reflected in the sophistication of the attacks.
- ▶ The end report issued by such organizations tends to not be political in nature. If the CEO has a wireless LAN access point in the office that compromises security, they will (usually) say so. A previous head of the CIA almost went to jail because an FBI report noted that he took his laptop (full of secrets) home on weekends and plugged it into his home network which was connected to the Internet. That is the sort of report that one is likely to get: a truthful one.
- ▶ Unlike many consulting organizations that wish to bury themselves deeply in an enterprise and then install an army of billable consultants, security organizations tend to not want this. They arrive, mount an attack and leave.
- ▶ Secrecy: the report prepared for management is usually kept very confidential. This means that management can fill security breaches as finances dictate and that it can do so without broadcasting weaknesses.

Using the same methods as attackers, your security policy and policy implementation can be tested. This is an opportunity to test the response of attack policies and procedures and see their effectiveness.

Attack types

There are two main types of attacks. The first is an attack where the attacker knows a considerable amount of information about the enterprise. The second is an attack where nothing is known. The general view is that if your enterprise can

withstand an “insider” attack (an attack where a considerable amount of information is known), then it is likely to withstand an external attack, with the possible exception of Denial Of Service (DOS) type of attacks which are really in a category of their own.

The methodology used to conduct planned attacks is similar to the method used by attackers, the exception being that a planned attack will not cause sustained damage to the enterprise. However, enterprise management should expect to have a plan for recovery from successful planned attacks. It is worth the inconvenience.

6.2 Evolution

All systems change and evolve. As systems evolve, threat profiles change and as these profiles change, so do levels of risk.

It is important to realize that the management of a security infrastructure is an ongoing issue. You cannot implement a security solution, validate it and walk away from it. You will need to keep the threat profile, the policy and the implementation current. This is at the heart of managing the security infrastructure. It requires commitment, money and time, all of which are valuable commodities in most enterprises.

6.2.1 PKI certificate management

One of the issues with any implementation of a PKI is certificate management.

WebSphere MQ makes use of PKI to implement SSL for channels. In order to do this, there is a public and private key pair associated with the queue manager and associated certificate(s). Certificates are typically valid for one year. If a certificate is allowed to expire, then channels using SSL will refuse to start. To obviate this possibility, management procedures should be developed to refresh certificates long before they expire; a common practice should be to create two year certificates that are replaced annually.

A certificate can be revoked. WebSphere MQ provides the ability to check certificates for validity by looking for a match in a certificate revocation list maintained in a LDAP directory. A certificate, for example, could be revoked if the private key of a queue manager were to become compromised.

Certificate Revocation Lists need to be managed in the LDAP directory and if a certificate is revoked, steps must be taken to issue a new certificate. If a certificate was revoked and a queue manager attempted to use that certificate in the SSL handshake, any secured channels would refuse to start.

CRLs are very useful for large implementations, where for example, hundreds or thousands of WebSphere MQ clients attach to queue managers. However, for smaller installations, it is almost always easier to simply remove the offending certificate from the certificate repository, rather than have to invoke a process to place an entry in the CRL and to then refresh the cached LDAP entries.

It is for this simple management reason that we elected in our business case to forgo implementing LDAP and CRL checking.

6.3 Management concerns summary

The following list summarizes the management tasks that are inherent in implementing a PKI and in implementing sound security policies and practices. As we will continue to see in this redbook, enterprise security encompasses far more than the installation of a few products and some software integration to tie them all together.

- ▶ Creating the equivalent of a DR plan for security attacks.
- ▶ Exercising such a plan on a regular basis.
- ▶ Managing certificates (this includes creation, revocation, renewals).
- ▶ Managing keys (this includes key creation, key strength, key algorithms, key reuse).
- ▶ Certificate revocation policy creation.
- ▶ Planning attack scenarios.
- ▶ Staffing.
- ▶ Cost management.

A most important factor is ensuring close cooperation with existing security administrators, hardware and network administrators, and, perhaps most importantly, ensuring that audit and senior management understand the need to implement enterprise wide security.



Business scenario

We anticipate that many WebSphere MQ customers will wish to exploit the new security capabilities first shipped with WebSphere MQ V5.3.

What this means in practice is that most readers of this redbook will be attempting to deploy features such as SSL to an already running production system. Recognizing this, we have therefore created a business scenario that we believe covers many customer installations.

We have created a mythical bank which we have called the Dolphin Bank; it has implemented a three tier WebSphere MQ architecture. The top two tiers of the architecture are located at the head office, and each branch office has a single Windows server supporting multiple WebSphere MQ clients.

Because it is a global enterprise, the bank has offices is several locations around the globe. We have elected to have these branch offices known by their WebSphere MQ cluster names.

The top tier is composed of three zSeries Logical Partitions (LPARs) running z/OS in a parallel sysplex. These are all members of a queue sharing group to ensure software fault tolerance and to ease maintenance concerns.

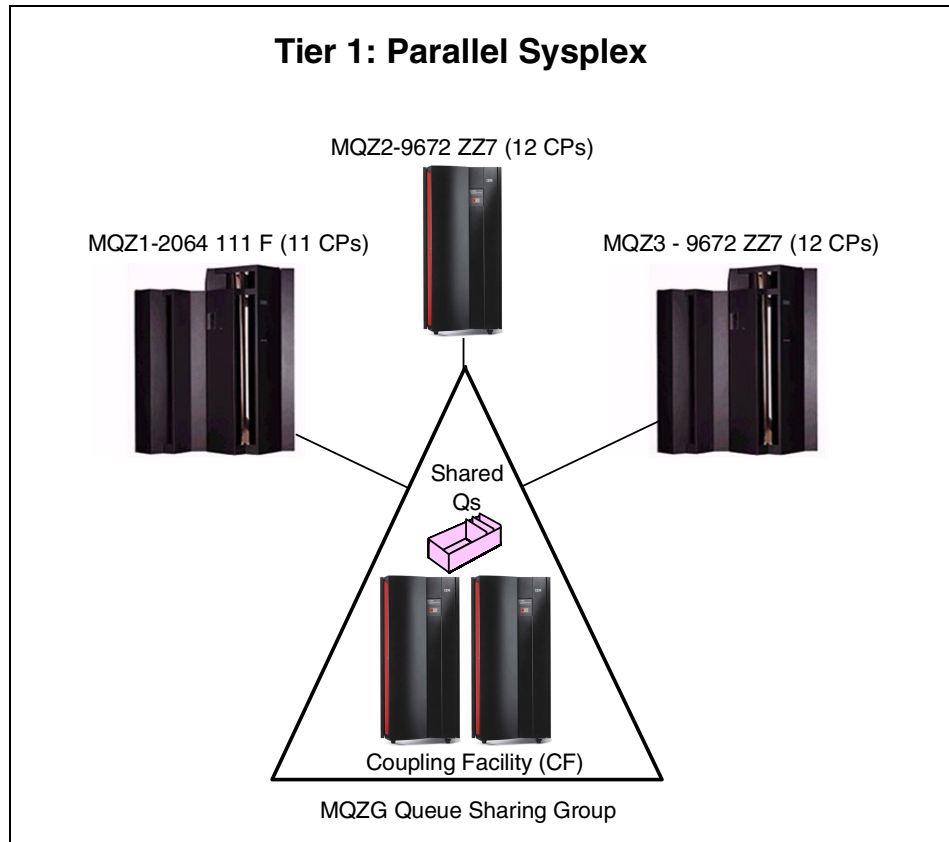


Figure 7-1 Top tier: parallel sysplex

The mid tier is composed of two pSeries machines running IBM AIX. These machines act as concentrators for WebSphere MQ traffic and are a buffer between the zSeries machine and the lower tier of xSeries machines running Windows 2000.

The lower tier of machines is used by bank employees. These are currently running WebSphere MQ clients for cost reasons. All of the machines in each branch are connected to a single WebSphere MQ server located in the branch.

Staff at head quarters are experimenting with Java technology and are attempting to connect programs written in both C and Java directly to z/OS using the WebSphere MQ client support in Windows.

7.1 Business to business

The Dolphin bank has a business to business connection with the Shetland Bank for global clearing services.

Over this dedicated link, a file of portfolio data is transferred every night. The file size is approximately 100 megabytes in size.

The management of the Dolphin Bank wishes to reduce costs.

They have therefore asked the WebSphere MQ infrastructure group to change the existing file transfer technique to use WebSphere MQ and to send the file over the Internet, secured with SSL. It is expected that the savings from removing the dedicated line between the banks will more than pay for the costs associated with the MQ SSL project in its entirety.

The Shetland bank uses an iSeries machine running OS/400 as its gateway to the Dolphin Bank and has WebSphere MQ installed on this machine in addition to the File Transfer Protocol (FTP) software currently used. Since the existing FTP transfer sometimes fails, both banks are looking forward to saving money and improving reliability.

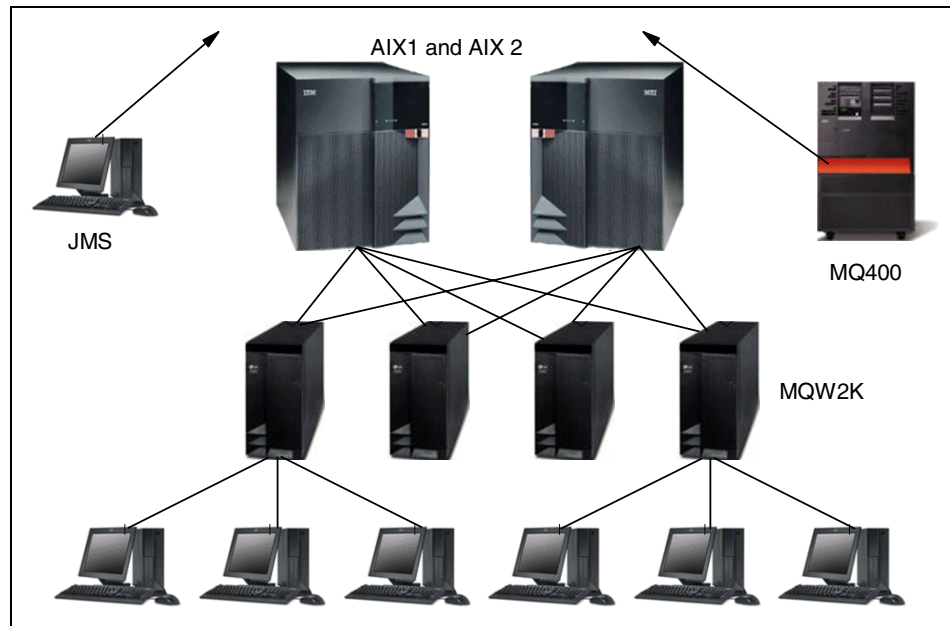


Figure 7-2 Mid and lower tiers

7.1.1 Environment summary

The Dolphin Bank has:

- ▶ Three LPARS in a parallel sysplex, containing MQZ1, MQZ2, MQZ3 in a queue sharing group named MQZG.
- ▶ Two AIX message concentrator machines MQAIX2, MQAIX3.
- ▶ Many Windows branch machines called MQW2K n (where n denotes a number) but we only reference two representative ones in our environment.
- ▶ Many Windows clients in each branch.
- ▶ An advanced technology group investigating Java and JMS.

The Shetland Bank has:

- ▶ An iSeries machine acting as a gateway.



Business scenario architecture

Our bank has grown through acquisitions. As it grew, it inherited various IT organizations that were more cost effective to integrate than to liquidate. Our bank elected to use WebSphere MQ as the “glue” to meld these organizations into the enterprise. This chapter looks at what the bank currently recognizes as its core systems and details the WebSphere MQ object definitions that hold them together.

8.1 Overview

Dolphin Bank Internationals system comprises the following:

- ▶ zSeries
- ▶ iSeries
- ▶ pSeries
- ▶ xSeries

These are configured as shown in Figure 8-1 on page 145.

8.1.1 Dolphin Bank International system configuration

The enterprise architecture of Dolphin Bank International is detailed in Figure 8-1 on page 145. The top tier consists of three queue managers MQZ1, MQZ2, MQZ3 in a queue sharing group. The queue sharing group is connected to the mid layer by two overlapping clusters called CCLUS1 and ECLUS1. Queues in the queue sharing group are clustered in either CCLUS1 or ECLUS1 but not both.

Multiple overlapping clusters exist to give a granular class of service. The reasons for this are discussed in “Cluster class of service” on page 149.

The mid tier acts as a concentrator for branch traffic. This tier consists of two queue managers, MQAIX2 and MQAIX3. These two queue managers are repositories for four overlapping clusters. The additional clusters are CCLUS2 and ECLUS2. The design of the clusters ensures that messages destined for the top tier are load balanced between MQAIX2 and MQAIX3. The way this is done is by creating alias objects with the same name shared in the CCLUS2 or ECLUS2 clusters on both queue manager pointing at objects shared in ECLUS1 or CCLUS1. Note that objects are only shared in one cluster and never more than one.

Restriction: There is an issue with WebSphere MQ clusters and cluster gateways. Alias names shared in a cluster should not be the same as the destination queue. If the alias names are the same as the target queue then there is a possibility that messages may bounce between alias queues before reaching the destination queue. The more alias queues shared in the cluster with the same name as the destination queue, the longer the message will take to reach the destination queue. This is because clustering makes no distinction between an alias queue, on a remote queue manager shared in a cluster, and a local queue, on a remote queue manager shared in a cluster. Therefore under the load balancing algorithm it is equally likely that a message could be sent to an alias queue as it is to a real queue.

The main model for messages is a request reply model. Therefore messages arriving at the top tier need to be able to flow back to the client machines in the branch offices. Since we have multiple clusters organized in classes of service, we wish to control the path that messages flow back. We wish to ensure that if a message flows up using one class of service, it returns using the same class of service. The way this is done is discussed in “An example message flow step-through” on page 159. However this materializes as a number of queue manager aliases that are defined on both AIX queue managers and shared in the appropriate cluster. The reasons behind the naming is also discussed later in this chapter.

The object definitions for these clusters on AIX are given in “Cluster object configuration” on page 150.

The bottom tier consists of Windows servers running in branch offices. WebSphere MQ client is used on branch workstations to connect to the branch office MQ servers. For the purpose of our case study, we have two branch queue managers called MQW2K2 and MQW2K5. Both of these queue managers are members of the CCLUS2 and the ECLUS2 clusters. The queue managers have local reply-to-queues defined that are not shared in the cluster for each application. There are four queue manager alias definitions that point to the local queue manager, shared in the appropriate cluster. Once again, this is to ensure that reply-to-messages flow back to the local queue manager using a known class of service. The naming of these queue manager aliases is discussed in “An example message flow step-through” on page 159. Clients connect to the queue manager by using a channel definition table. The configuration of this definition table is such that if the local queue manager in the branch office is not available, a connection will be attempted to another branch office queue manager. The server connection channels have the MCAUSER parameter set.

The transfer between iSeries and zSeries is initiated by a batch file to the message program on z/OS. When the message is received on OS/400, a CL program is triggered. The CL program in Appendix B on page 265 gets the message from the queue and puts a response using the sample programs. The original configuration was implemented on clear channels (CC in the configuration) and, when SSL became available, an additional pair of channels (SC for secure) was added.

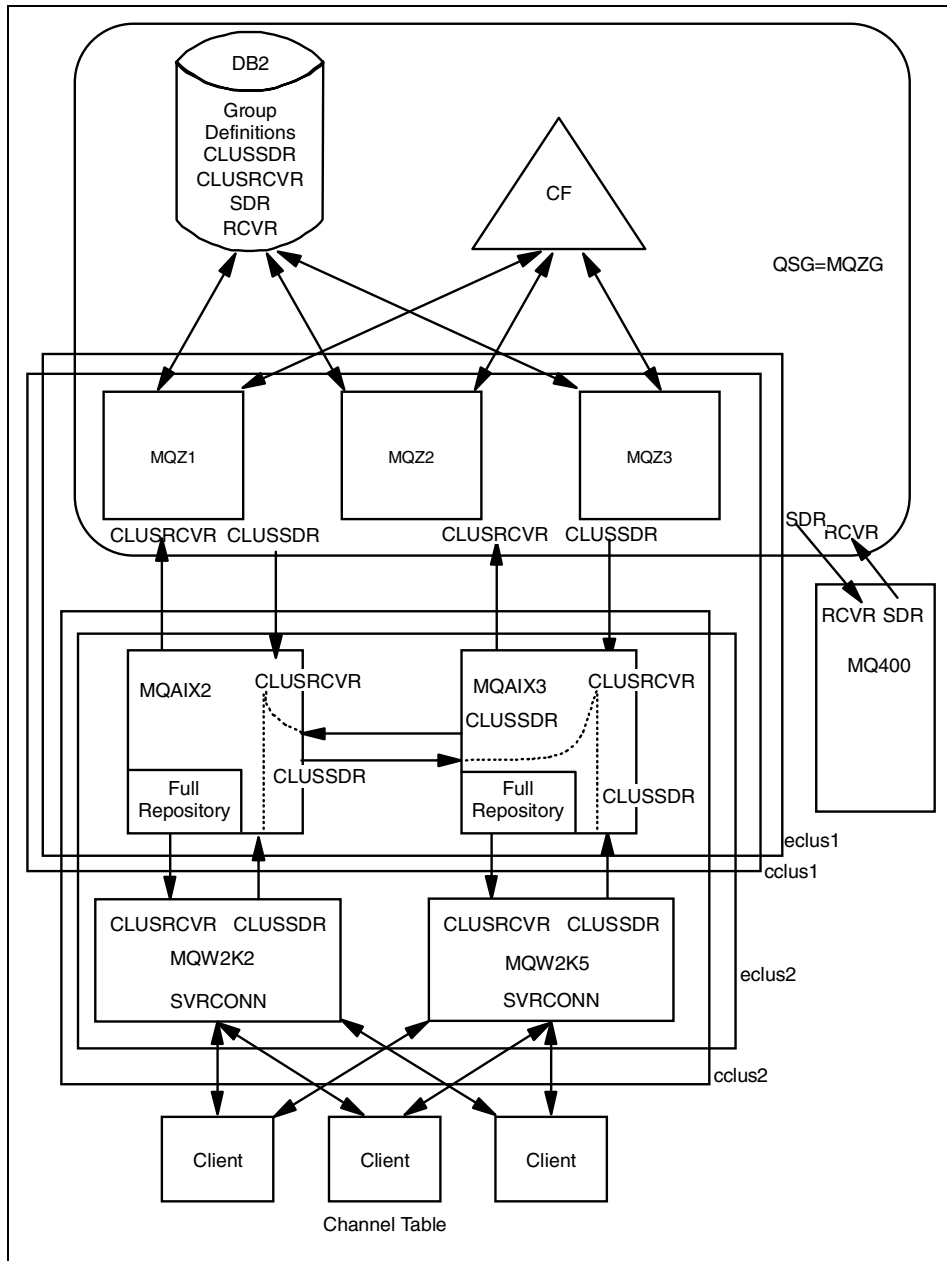


Figure 8-1 Dolphin Bank International configuration

8.2 Hardware and software

The following sections describe the hardware levels and the software and object definitions required on each platform in our business case to obtain the configuration in Figure 8-1 on page 145. At least these levels are required if you replicate any configuration described in this section and, ideally, use the current CSD level when you are ready to undertake your own project. A discussion of the message flow and the classes of service is also presented.

8.2.1 zSeries

zSeries is the top tier in the configuration in our business case scenario.

Hardware

The zSeries component of the system is part of a parallel sysplex that runs multiple z/OS images (LPARs running across two boxes, 9672-X17 and a 2064-IC7), and two CF (Coupling Facility) images (one on each box).

The setup at the Dolphin Bank comprises three systems: SC61, SC62 and SC66. These are spread across two LPARs, SC61 and SC62 being on LPAR1 (on the 9672) and SC66 being on LPAR2 (on the 2064).

Software

- ▶ SC61 and SC62 running z/OS V1.1
- ▶ SC66 running z/OS V1.2.
- ▶ Coupling Facility levels:
 - CF05 running CFLEVEL 9 CFCC RELEASE 09.00 service level 01.15
 - CF06 running CFLEVEL 12 CFCC RELEASE 12.00 service level 04.15
- ▶ WebSphere MQ V5.3 with the service levels available in mid-October 2002
- ▶ DB2 V7
- ▶ CICS : CICSTS V2.2

Queue manager

There are three queue managers:

- ▶ MQZ1MSTR
- ▶ MQZ2MSTR
- ▶ MQZ3MSTR

These are part of a queue sharing group, MQZG. Each queue manager has a channel initiator (mover) with it, respectively:

- ▶ MQZ1CHIN
- ▶ MQZ2CHIN
- ▶ MQZ3CHIN

There is a shared listener listening on port 1540 which interfaces to the queue sharing group.

Each queue manager has a queue manager listener, respectively:

- ▶ MQZ1 listens on lpaddr (9.12.6.106) port (1543)
- ▶ MQZ2 listens on lpaddr (9.12.6.107) port (1541)
- ▶ MQZ3 listens on lpaddr (9.12.6.108) port (1542)

The data sharing group is DB72U.

The diagram in Figure 8-2 shows the z/OS setup.

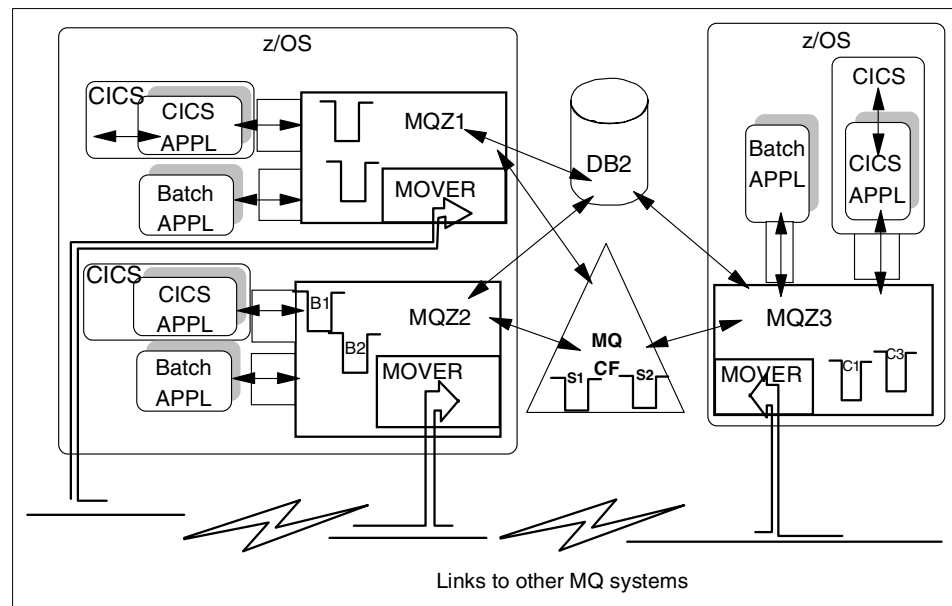


Figure 8-2 z/OS setup

8.2.2 pSeries

pSeries is the middle tier concentrator between the mainframe z/OS system and branch office Windows servers. We have used two pSeries servers for workload balancing and continuous operations using WebSphere MQ clustering.

Hardware

- ▶ RS6000 44P Model 270
- ▶ RS6000 F50

Software

- ▶ Operating system: AIX V5.1
- ▶ WebSphere MQ V5.3 with CSD1

Queue manager

There is one queue manager on each machine, respectively:

- ▶ MQAIX2
- ▶ MQAIX3

8.2.3 xSeries

xSeries is the lower tier that houses the branch office servers.

Hardware

- ▶ IBM NetVista™ Personal Computers

Software

- ▶ Operating system: Windows 2000 with SP3
- ▶ WebSphere MQ V5.3 with CSD1

Queue manager

- ▶ MQW2K2
- ▶ MQW2k5

8.2.4 iSeries

iSeries is an additional tier connected to the z/OS system.

Hardware

- ▶ iSeries 9406 Model 170

Software

- ▶ Operating system: OS/400 V5r1
- ▶ WebSphere MQ V5.3

Queue manager

There is one queue manager: MQ400

8.3 Transactions from Windows through AIX to z/OS

This section discusses the core systems and details of the WebSphere MQ object definitions that hold them together.

8.3.1 Cluster class of service

In the business scenario, the clustered environment has been organized into classes of service. These classes of service are denoted as the C (clear) class and the E (encrypted) class. The naming of these classes is therefore CCLUS n and ECLUS n . The design goals did not permit the queues on the top tier to be directly visible in the branches. Therefore, four overlapping clusters have been created. The middle tier acts as the gateway between the top two clusters and the bottom two clusters. Queues in the top two clusters are only visible to the bottom tier by aliases on the middle tier.

The technical driver for different classes of service is to ensure that granular controls can be implemented. For example, if different message sizes flow through the clusters infrequently while small messages flow frequently, classes of service can be used to separate the transactions into separate clusters. One cluster can be tuned for small, frequent messages whilst another can be tuned for large messages. Another reason for this could be for different network protocols, or secured network and public network, for example.

In our business scenario, the driver for the classes of service was to introduce security services on channels. Using the different classes of service, we are able to provide different levels of security on channels. The C class cluster is designed for messages to be sent in clear and the E class is intended for

messages to be enciphered. However, it is quite possible that this sort of setup exists in organizations for other reasons, where security has not been considered.

To achieve different classes of service, queue managers may be in one or more clusters. However, the queue manager must have a cluster receiver channel for each cluster, that is, shared only in that cluster. This affords control over the flow of messages between clusters.

Consider two queue managers, QMA and QMB. Both of the queue managers are repositories for CLUS1 and CLUS2. QMA and QMB have two cluster sender and two cluster receiver channels. These channels are only shared in one cluster.

For example, one of the cluster receiver channels on QMA may have the following definition:

```
def chl(TO.CCLUS1.QMA) chltype(CLUSRCVR) cluster(CLUS1)
```

If a queue is shared only in CLUS1 and not in both clusters, we can control the channel through which a message will flow between the queue managers in that cluster.

8.3.2 Cluster object configuration

There are, physically, two level of networks in this system.

1. Intranet level

There are sysplex systems on zSeries, two pSeries servers and one iSeries of Shetland Bank in the Dolphin Bank's private network.

2. Internet level

There are some Windows servers at branch offices and Windows client machines.

There are four MQ clusters:

- ▶ CCLUS1 with members:

- MQZ1
- MQZ2
- MQZ3
- MQAIX2 (full repository)
- MQAIX3 (full repository)

- ▶ ECLUS1 with members:

- MQZ1
- MQZ2
- MQZ3

- MQAIX2 (full repository)
- MQAIX3 (full repository)
- ▶ CCLUS2 with members:
 - MQAIX2 (full repository)
 - MQAIX3 (full repository)
 - MQW2K2
 - MQW2K5
- ▶ ECLUS2 with members:
 - MQAIX2 (full repository)
 - MQAIX3 (full repository)
 - MQW2K2
 - MQW2K5

Important: Clusters must have at least one full repository. However, it is recommended that you have two full repositories. Only in rare circumstances would you need more than two full repositories.

Naming standards for channels are defined as:

TO.CLUSTERNAME.QUEUEMANAGERNAME.

Configuration details

First, we configured full repositories for each cluster.

- ▶ On MQAIX2
 - a. Create a name list to join multiple clusters.


```
DEFINE NAMELIST(DOLPHIN) NAMES(ECLUS1,CCLUS1,ECLUS2,CCLUS2)
```
 - b. Alter the queue managers to be full repositories.


```
ALTER QMGR REPOSNL(DOLPHIN)
```
 - c. Define a cluster-receiver channel for each cluster.


```
DEFINE CHANNEL(TO.ECLUS2.MQAIX2)          +
          CHLTYPE(CLUSRCVR) TRPTYPE(TCP)    +
          CONNAME('9.20.18.71(1414)')      +
          CLUSTER(ECLUS2)

DEFINE CHANNEL(TO.CCLUS2.MQAIX2)          +
          CHLTYPE(CLUSRCVR) TRPTYPE(TCP)    +
          CONNAME('9.20.18.71(1414)')      +
          CLUSTER(CCLUS2)

DEFINE CHANNEL(TO.ECLUS1.MQAIX2)          +
          CHLTYPE(CLUSRCVR) TRPTYPE(TCP)    +
```

```
CONNAME('9.20.18.71(1414)') +  
CLUSTER(ECLUS1)
```

```
DEFINE CHANNEL(TO.CCLUS1.MQAI2) +  
CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +  
CONNAME('9.20.18.71(1414)') +  
CLUSTER(CCLUS1)
```

– Define Cluster-sender channels to the other full repository, MQAIX3.

```
DEFINE CHANNEL(TO.ECLUS2.MQAI3) +  
CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
CONNAME('9.20.18.175(1414)') +  
CLUSTER(ECLUS2)
```

```
DEFINE CHANNEL(TO.CCLUS2.MQAI3) +  
CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
CONNAME('9.20.18.175(1414)') +  
CLUSTER(CCLUS2)
```

```
DEFINE CHANNEL(TO.ECLUS1.MQAI3) +  
CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
CONNAME('9.20.18.175(1414)') +  
CLUSTER(ECLUS1)
```

```
DEFINE CHANNEL(TO.CCLUS1.MQAI3) +  
CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
CONNAME('9.20.18.175(1414)') +  
CLUSTER(CCLUS1)
```

d. Then configure a partial repository to join the cluster.

► On Windows (the example given is for MQW2K2)

a. Define cluster-sender and cluster-receiver to join CCLUS2

```
DEFINE CHANNEL(TO.CCLUS2.MQW2K2) +  
CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +  
CONNAME('9.20.31.89(1414)') +  
CLUSTER(CCLUS2)
```

```
DEFINE CHANNEL(TO.CCLUS2.MQAI2) +  
CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
CONNAME('9.20.18.71(1414)') +  
CLUSTER(CCLUS2)
```

b. Define cluster-sender and cluster-receiver to join ECLUS2

```
DEFINE CHANNEL(TO.ECLUS2.MQW2K2) +  
CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +  
CONNAME('9.20.31.89(1414)') +  
CLUSTER(ECLUS2)
```

```

DEFINE CHANNEL(TO.ECLUS2.MQAI2)+
        CHLTYPE(CLUSSDR) TRPTYPE(TCP)+
        CONNAME('9.20.18.71(1414)')+
        CLUSTER(ECLUS2)

```

► On z/OS

- a. Define cluster-receivers on each queue manager to join CCLUS1 and ECLUS1. The cluster-receivers must be queue manager objects and have local IPADDRs and local ports.

For MQZ1:

```

DEFINE CHANNEL(TO.CCLUS1.MQZ1) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.12.6.106(1543)') +
        CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQZ1) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.12.6.106(1543)') +
        CLUSTER(ECLUS1)

```

- b. Define the following cluster-sender channels on each queue manager to join CCLUS1 and ECLUS1:

```

DEFINE CHANNEL(TO.CCLUS1.MQAI2) +
        CHLTYPE(CLUSSDR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.20.18.71(1414)') +
        CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQAI2) +
        CHLTYPE(CLUSSDR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.20.18.71(1414)') +
        CLUSTER(ECLUS1)

```

8.3.3 Queues

There are two simple message flows.

Request applications put a request message into a queue and wait for the response from the queue. The request message contains ReplyToQueue and ReplyToQMGr (which should not be left to default) fields in MQMD.

On z/OS, response applications wait for a request message and put their response message into the queue specified in ReplyToQueue of MQMD.

Request flow

WebSphere MQ Client connects to the branch queue manager and puts a message to the request queue specifying the ReplyToQueue and ReplyToQMGr. The AIX queue managers act as the gateway to z/OS sysplex which has the real request queues.

Table 8-1 Request flow

Queue alias on AIX	Target shared queue on z/OS
ACCOUNT.REQUEST.EC	ACCOUNT.REQUEST.EC.CICS
PORTFOLIO.REQUEST.EC	PORTFOLIO.REQUEST.EC.CICS
ATM.REQUEST.CC	ATM.REQUEST.CC.CICS

Since these three request queues are alias queues shared either in the CCLUS2 or ECLUS2 cluster on the AIX queue managers, no definitions are required on Windows. Their target queue for the alias queues are the shared queues on z/OS shared in either CCLUS1 or ECLUS1.

Important: You have to define the Alias queue's DEFBIND attribute to (NOTFIXED) for alias queues to act as a gateway between clusters

Example definition for ACCOUNT.REQUEST.EC:

► On Windows

Nothing to define for request.

► On AIX (both queue managers)

```
DEFINE QALIAS(ACCOUNT.REQUEST.EC)      +
      TARGQ(ACCOUNT.REQUEST.EC.CICS)   +
      DEFBIND(NOTFIXED)                 +
      CLUSTER(ECLUS2)
```

► On z/OS

```
DEFINE QLOCAL(ACCOUNT.REQUEST.EC.CICS) +
      CFSTRUCT(APPLICATION1)           +
      QSGDISP(SHARED)                   +
      DEFSOPTS(SHARED)                   +
      CLUSTER(ECLUS1)                     +
      SHARE)
```

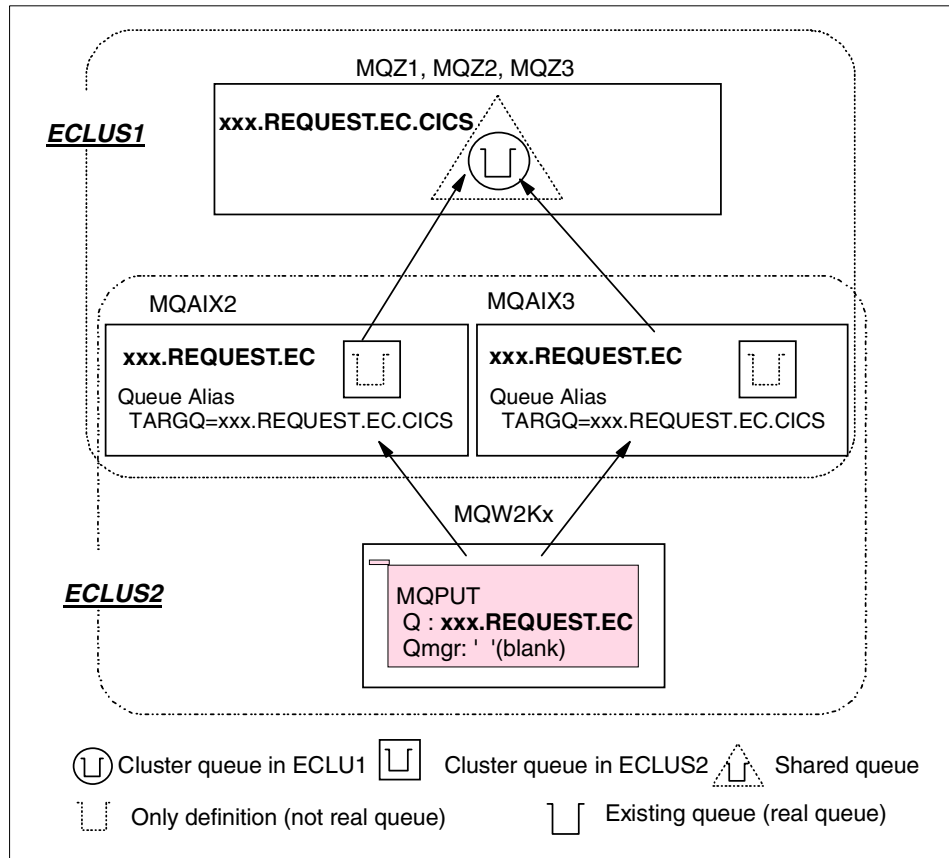


Figure 8-3 Request queue definition

Response flow

A CICS program processes the response to the request. When the program opens a queue for response, it should set appropriate ObjectName and ObjectQmgrName in MQOD (MQ Object Descriptor). These are determined by the request MQMD fields.

Table 8-2 Response flow

Request MQMD	MQOD
ReplyToQueue	ObjectName
ReplyToQmgr	ObjectQmgrName

AIX queue managers are also the gateway to the Windows queue managers which contain the response queues.

In order to resolve the response queue manager name, we used queue manager alias definitions on AIX and Windows.

We wanted to maintain the class of service for messages on the return path. Therefore, if messages flow out with a C class, they should return using the C class.

The requesting application on Windows specifies a different queue manager name from the local queue manager for the ReplyToQmgr in the MQMD field. The naming standard for this is qmgrCC for the C class and qmgrEC for the E class. We used queue manager aliases to resolve the queue manager name to enable us to control the return path (for example, MQW2K2EC for an E class message and MQW2K2CC for a C class message).

As illustrated in the tables below, the queue manager aliases defined on the AIX queue managers are shared in the CCLUS1 or ECLUS1 clusters. These aliases, in turn, point to queue manager aliases on the Windows queue managers, which resolve to the local queue manager.

Table 8-3 Queue manager aliases resolution AIX2

AIX2	Windows	
MQW2KnEC	MQW2KnECA	MQW2Kn
MQW2KnCC	MQW2KnCCA	

Table 8-4 Queue manager aliases resolution AIX3

AIX3	Windows	
MQW2KnEC	MQW2KnECB	MQW2Kn
MQW2KnCC	MQW2KnCCB	

Restriction: The design of the reply path was created in this way because we wanted to ensure that the reply messages maintained the class of service. However, this is a potential problem that can be encountered. It is possible for messages to bounce between queue managers when queue manager aliases are named the same as the RQMNAME.

For example, if we created a queue manager alias named MQW2K5EC on MQAIX2 and MQAIX3 (shared in ECLUS1) where the RQMNAME was also MQW2K5EC, and defined a queue manager alias on MQW2K5 called MQW2K5EC (shared in ECLUS2) with an RQMNAME of MQW2K5, messages would have bounced between the MQAIX2 and MQAIX3 before arriving at MQW2K5.

To some extent, our design overcompensated for this by ensuring that the queue manager aliases on the AIX queue managers resolved to different queue manager aliases on the Windows queue managers and by having two queue manager aliases pointing to the local queue manager on the Windows queue managers. Another option would be to have only one queue manager alias on the Windows queue managers and point both queue manager aliases on the AIX queue managers to resolve to that one alias. This should also work in a satisfactory manner.

Queue manager aliases in a cluster behave much the same as queue aliases with the restriction discussed in “Dolphin Bank International system configuration” on page 142.

Example definition for ACCOUNT.RESPONSE.EC at MQW2K1

► On z/OS

Nothing to define for response.

► On AIX2

```
DEFINE QREMOTE(MQW2K1EC)      +
      RQMNAME(MQW2K1ECA)     +
      DEFBIND(NOTFIXED)      +
      CLUSTER(ECLUS1)
```

► On AIX3

```
DEFINE QREMOTE(MQW2K1EC)      +
      RQMNAME(MQW2K1ECB)     +
      DEFBIND(NOTFIXED)      +
      CLUSTER(ECLUS1)
```

► On Windows

```
DEFINE QREMOTE(MQW2K1ECA) +
      RQMNAME(MQW2K1) +
      DEFBIND(NOTFIXED) +
      CLUSTER(ECLUS1)
```

```
DEFINE QREMOTE(MQW2K1ECB) +
      RQMNAME(MQW2K1) +
      DEFBIND(NOTFIXED) +
      CLUSTER(ECLUS1)
```

The local queue for the response message is DEFINE QLOCAL(ACCOUNT.RESPONSE.EC).

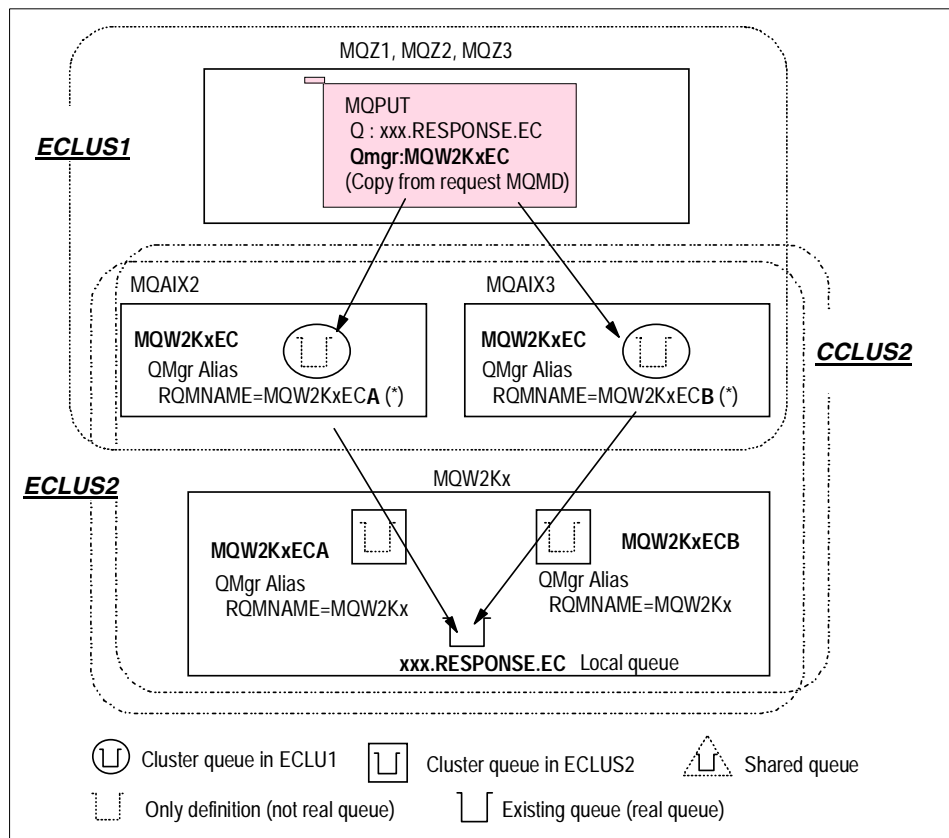


Figure 8-4 Response queue definition

Note: In our business scenario, we have defined separate queue managers to maintain the class of service between ECLUS2 and CCLUS2.

8.3.4 Applications

This section discusses various applications and their functions in our business scenario.

The ATM application

The ATM application is an enquiry application. Customers can SMS their branch from any location, stating their location; the branch will be able to give them a list of local ATMs. The information, therefore, is of low value from a security standpoint.

The portfolio application

The portfolio application is designed so that a customer can walk into a branch of the Dolphin Bank anywhere in the world and get a valuation of their portfolio of shares at that moment in time. The data contains confidential information and may influence a customer to buy or sell shares.

The account application

The account application is a balance enquiry and transfer application. The information is confidential.

8.3.5 An example message flow step-through

This section describes the message flow within our environment.

The ATM message flow

The flow is as follows:

1. The client requests that an ATM be found by city or zip code.
2. A request is generated and put to the ATM.REQUEST.CC queue on queue manager MQW2Kn, where n denotes a serial number. For the sake of this example, we will use MQW2K5. ATM.REQUEST.CC is an alias queue shared in the CCLUS2 cluster on both AIX queue managers with DEFBIND(NOTFIXED). The request is put specifying the reply-to-queue to be ATM.RESPONSE.CC and the reply-to-queue manager to be MQW2K5CC.
3. The message will flow to one of the AIX queue managers. For the example, we will consider that the message has moved to MQAIX2 on channel TO.CCLUS2.MQAIX2.
4. The MQAIX2 resolves the QALIAS to ATM.REQUEST.CC.CICS shared in CCLUS1 on the queue sharing group.

5. The message is processed by a queue manager in the queue sharing group and a reply is put to the reply-to-queue manager MQW2K5CC with the destination queue to be ATM.RESPONSE.CC.
6. MQW2K5CC is resolved using a queue manager alias with DEFBIND(NOTFIXED) on MQAIX2 and MQAIX3 that is shared in the cluster CCLUS1.
7. The message flows to either MQAIX2 or MQAIX3 (using the round robin algorithm to decide on the queue manager) down a channel that is available to the CCLUS1 cluster. For this example, we will consider that the message has arrived at MQAIX3.
8. The queue manager is resolved from MQW2K5CC to MQW2K5CCB. MQW2K5CCB is a queue manager alias shared in the cluster CCLUS2 on MQW2K5. Therefore, the message flows to MQW2K5 on a channel that is in the CCLUS2 cluster.
9. The queue manager alias is resolved to the local queue manager MQW2K5 and the message is put to the response queue.

8.4 File transfer from z/OS to OS/400

In our environment, the bank needs to send a batch file on a nightly basis from z/OS to the portfolio system on OS/400. The file needs to be sent as a sequence of messages to ensure ease of recovery from occasional line failures and the data needs to be signed and encrypted.

This portfolio data file transfer is initiated from the z/OS side of the connection.

To accomplish this, we used a file transfer program initially developed by IBM and shipped with early versions of MQSeries as an unsupported sample. The program reads a physical sequential dataset and sends it as a series of messages followed by a null record (used to indicate end of file). The program was altered to allow it to write to remote queue definitions.

When compiling programs, we have often found that customers rely on catalogued procedures and second-hand Job Control Language (JCL) that is time consuming to correctly modify. We have therefore included the JCL needed to assemble and link this program following the program source, along with the JCL needed to execute the program. Refer to Appendix B on page 265 for the sample JCL.

This program will be used in subsequent chapters of this book to illustrate uses for the ALTERNATE USER ID field in the MQMD (MQ Message Descriptor).

8.4.1 OS/400 configuration

WebSphere MQ for iSeries Quick Beginnings provides detailed guidance to get started.

For your convenience, below is a fast track post WebSphere MQ configuration guide, using qsecofr or a user with sufficient access rights.

- ▶ `strsbs qmqm/qmqm` (starts the WebSphere MQ subsystem)

```
MAIN                               AS/400 Main Menu                               System:  ITS0400
Select one of the following:
    1. User tasks
    2. Office tasks
    3. General system tasks
    4. Files, libraries, and folders
    5. Programming
    6. Communications
    7. Define or change the system
    8. Problem handling
    9. Display a menu
   10. Information Assistant options
   11. Client Access/400 tasks

   90. Sign off

Selection or command
==> strsbs qmqm/qmqm
-----
F3=Exit  F4=Prompt  F9=Retrieve  F12=Cancel  F13=Information Assistant
F23=Set initial menu
```

Figure 8-5 Main menu

- ▶ `wrkmqm` (lists queue managers defined and allows start and stop access to queues and channels)

```

Work with queue managers

Type options, press Enter.
 2=Change  4=Delete  5=Display  14=Start  15=End  18=Work with Queues
 19=Work with Processes  20=Work with Channels ...

Opt  Name                               Status  Default
---  ---                               ---     ---
---  MQ400                               ACTIVE  NO
---  TEST                                INACTIVE NO

                                           Bottom

Parameters or command
===>
F3=Exit  F4=Prompt  F5=Refresh  F6=Create  F9=Retrieve  F12=Cancel
F16=Repeat position to  F17=Position to  F23=More options  F24=More keys

```

Figure 8-6 Work with queue manager

- ▶ `crtmqm`, `strmqm` and others available at the command line. Parameters can be found by entering the command and prompting using the **F4** key.

If you wish to remotely administer the queue manager from MQ Explorer:

1. From `wrkmqm`, tab to **qmgr** and enter 20 (work with channels).
2. Find `SYSTEM.DEFAULT.SVRCONN` and copy 3.

```

Work with MQ Channels

Queue Manager Name . . : MQ400

Type options, press Enter.
 2=Change  3=Copy  4=Delete  5=Display  8=Work with Status  13=Ping
14=Start  15=End  16=Reset  17=Resolve

Opt  Name                Type          Transport    Status
---  ---                ---          ---          ---
---  SYSTEM.DEF.CLNTCONN   *CLTCN      *TCP         INACTIVE
---  SYSTEM.DEF.CLUSRCVR   *CLUSRCVR   *TCP         INACTIVE
---  SYSTEM.DEF.CLUSSDR    *CLUSSDR    *TCP         INACTIVE
---  SYSTEM.DEF.RECEIVER   *RCVR       *TCP         INACTIVE
---  SYSTEM.DEF.REQUESTER *RQSTR      *TCP         INACTIVE
---  SYSTEM.DEF.SENDER     *SDR        *TCP         INACTIVE
---  SYSTEM.DEF.SERVER     *SVR        *TCP         INACTIVE
3    SYSTEM.DEF.SVRCONN   *SVRCN      *TCP         INACTIVE

Bottom

Parameters or command
===>
F3=Exit   F4=Prompt   F5=Refresh  F6=Create   F9=Retrieve  F12=Cancel
F16=Repeat position to  F17=Position to  F21=Print

```

Figure 8-7 Work with WebSphere MQ channels

- In the To channel field, enter SYSTEM.ADMIN.SVRCONN.

```

Copy MQ Channel (CPYMQMCHL)

Type choices, press Enter.

From channel . . . . . > 'SYSTEM.DEF.SVRCONN'
To channel . . . . . > 'SYSTEM.ADMIN.SVRCONN'
Message Queue Manager name . . . > 'MQ400'

Bottom

F3=Exit   F4=Prompt   F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys

```

Figure 8-8 Copy WebSphere MQ channel

- In the Message channel agent user ID field, enter QMQM.

```

Copy MQ Channel (CPYMQMCHL)

Type choices, press Enter.

From channel . . . . . > 'SYSTEM.DEF.SVRCONN'
To channel . . . . . > 'SYSTEM.ADMIN.SVRCONN'
Message Queue Manager name . . . > 'MQ400'

Channel type . . . . . > *SVRCN
Replace . . . . . *NO *NO, *YES
Transport type . . . . . *SAME *LU62, *TCP, *SAME
Text 'description' . . . . . *SAME

Transaction Program Name . . . . *SAME

Mode Name . . . . . *SAME Character value, *BLANK...
Message channel agent user ID . . QMQM Character value, *NONE...

Bottom
F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

Figure 8-9 MCA user ID

5. Enter `strmq*` at the command line to find the `STRMQMCSVR` command.
6. Select **1** and enter the `qmgr_name` name on the next screen, or enter `STRMQMCSVR qmgr_name` at the command line.

```

Select Command

Type options, press Enter.
1=Select

Opt Command Library Text
- STRMQM QSYS Start Message Queue Manager
- STRMQMCHL QSYS Start MQM Channel
- STRMQMCHLI QSYS Start MQM Channel Initiator
- STRMQMCSVR QSYS Start MQM Command Server
- STRMQMDLQ QSYS Start MQ DLQ Handler
- STRMQMLSR QSYS Start MQM Listener
- STRMQMMQSC QSYS Start MQM MQ Commands
- STRMQMTRM QSYS Start MQM Trigger Monitor
- STRMQM QMQM Start Message Queue Manager
- STRMQMCHL QMQM Start MQM Channel
- STRMQMCHLI QMQM Start MQM Channel Initiator
1 STRMQMCSVR QMQM Start MQM Command Server

Parameters or command
===> █

F3=Exit F4=Prompt F5=Refresh F9=Retrieve F11=Display names only
F12=Cancel F16=Repeat position to F17=Position to F24=More keys
More...

```

Figure 8-10 Start command server

7. Enter `STRMQMLSR MQMNAME(qmgr_name) PORT(nnnn)`; again `startmq*` can be used as a starting point.

The familiar WebSphere MQ Explorer GUI can now be used in Windows to add MQ objects and manage the queue manager from a central point. From a security point of view, it will be necessary to disable the SYSTEM.ADMIN.SVRCONN channel at a later stage.

We needed to add a default route to the TCP/IP to get to the gateway router using CFGTCP, 2. Find this from a machine on the same network, for example, ipconfig in a cmd window.

Channels could then be started and data flowed between the remote and local queues.



Business scenario security configuration

In this chapter, we describe the WebSphere MQ security that is currently deployed in our environment. We discuss the default situation that most customers inherit following an initial installation and continue to propagate through subsequent upgrades.

9.1 z/OS

The initial security setup on z/OS is very basic, and the security profiles defined do not provide much granularity. The majority of profiles defined are generic profiles, which means that it is very difficult to control access to all your different resources.

The CLISTs shown in Appendix B, “Scripts, samples code and JCL” on page 265 in the section headed “CLISTs for RACF definitions and access levels for business scenario” on page 276 show the definitions used and the access levels granted.

The receiver channel definitions have been set up with PUTAUT = DEF and the MCAUSER attribute left blank. This means that the only user ID that will be checked for any PUTs coming in across the mover will be the channel initiator address space user ID. This is not generally a good idea since this user ID, by its nature, tends to have high authority to most queues simply in order to function.

9.2 AIX

There is nothing special to mention for security setup on WebSphere MQ for AIX.

Object Authority Manager (OAM)

By default, OAM is turned on but object access control is not set for each queue.

Channel attribute PUTAUT

The receiver channel attribute is set with PUTAUT=DEF and MCAUSER=' '(blank).

This means that all messages coming through the channel are put to the queue by the MQ listener. It is always mqm.

9.3 Windows 2000

Following are notes on the setup for Windows 2000.

9.3.1 Client connections

In our business scenario, clients and servers are not in a domain. Each user ID has a SID associated as we have discussed in Chapter 4, “Platform security” on page 45. WebSphere MQ passes this SID information and each account is viewed as user@domain. Therefore, if user Bob is on client PC ITSOA, the user

account would be bob@itsoa with an associated SID. If the server were called ITSOZ then even with a local account called Bob, the connection would fail. An example of this error is given below:

```
24/10/2002 14:13:17
AMQ8074: Authorization failed as the SID
'S-1-5-21-329068152-1801674531-725345543-1004' does not match the entity
'bob'.
```

EXPLANATION:

The Object Authority Manager received inconsistent data - the supplied SID does not match that of the supplied entity information.

ACTION:

Ensure that the application is supplying valid entity and SID information.

To overcome this, a user called mquser1 was created on all WebSphere MQ Windows 2000 servers. A group called MQAPPS was also created. MCAUSER on all the server connection channels was set to mquser1. An example definition is given below:

```
DEFINE CHANNEL ('CLIENT.MQW2K5') CHLTYPE(SVRCONN) +
TRPTYPE(TCP) +
DESCR(' Branch Office Clients') +
MCAUSER('mquser1') +
REPLACE
```

9.3.2 The OAM settings

The OAM was then set to allow the group MQAPPS to connect to the queue manager, put to cluster queues, and then get from response queues. This was done in a simple script given below:

```
@echo off
REM -----
REM setmqaut usage: setmqaut -m QMgrName [-n ObjName] -t ObjType [-p
Principal | -g Group]
REM [-s ServiceName] Authorizations
REM -----
set QMGR=MQW2K5
setmqaut -m %QMGR% -t qmgr -g MQAPPS +connect
setmqaut -m %QMGR% -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t q -g MQAPPS -all
setmqaut -m %QMGR% -n **.RESPONSE.* -t q -g MQAPPS +get +browse
```

This gives all connections on the client channel the authority to connect to the queue manager, get from all the queues that have RESPONSE in their name and then put to all cluster queues. This is therefore quite open security.

9.4 OS/400

This section illustrates how to display and update object authorization on OS/400.

9.4.1 Overview

Entering WRKMQMAUT then pressing **F4** is an easy way to display existing authorizations on OS/400. WRKMQMAUT MQMNAME(MQ400) produces the same result.

Work with MQ Authority (WRKMQMAUT)		
Type choices, press Enter.		
Object/Profile name	*ALL	
Object type	*ALL	*Q, *PRC, *MQM, *NMLIST...
Output	*	*, *PRINT
Message Queue Manager name . . .	MQ400	
Bottom		
F3=Exit	F4=Prompt	F5=Refresh
F24=More keys	F12=Cancel	F13=How to use this display

Figure 9-1 Work with MQ authority

A group is defined for easy administration based on qmqmadm.

```

Display User Profile - Basic
User profile . . . . . : QMUSER
Previous sign-on . . . . . :
Sign-on attempts not valid . . . . . : 0
Status . . . . . : *ENABLED
Date password last changed . . . . . : 10/23/02
Password expiration interval . . . . . : *NOMAX
Set password to expired . . . . . : *NO
User class . . . . . : *USER
Special authority . . . . . : *NONE
Group profile . . . . . : *NONE
Owner . . . . . : *USRPRF
Group authority . . . . . : *NONE
Group authority type . . . . . : *PRIVATE
Supplemental groups . . . . . : *NONE
Assistance level . . . . . : *SYSVAL
Current library . . . . . : *CRTDFT
More...

Press Enter to continue.

F3=Exit F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2000.

```

Figure 9-2 Group profile

A user is defined who will get and put once MQ authorizations have been added.

```

Display User Profile - Basic
User profile . . . . . : BOB
Previous sign-on . . . . . : 10/23/02 10:09:56
Sign-on attempts not valid . . . . . : 0
Status . . . . . : *ENABLED
Date password last changed . . . . . : 10/23/02
Password expiration interval . . . . . : *NOMAX
Set password to expired . . . . . : *NO
User class . . . . . : *USER
Special authority . . . . . : *JOBCTL
Group profile . . . . . : QMUSER
Owner . . . . . : *USRPRF
Group authority . . . . . : *ALL
Group authority type . . . . . : *PGP
Supplemental groups . . . . . : *NONE
Assistance level . . . . . : *SYSVAL
Current library . . . . . : *CRTDFT
More...

Press Enter to continue.

F3=Exit F12=Cancel
(C) COPYRIGHT IBM CORP. 1980, 2000.

```

Figure 9-3 User profile

Authority is granted to Bob's group QMUSER.

```

Grant MQ Object Authority (GRMQMAUT)

Type choices, press Enter.

Object name . . . . . > MQ400
-----
Object type . . . . . > *MQM          *ALL, *Q, *ALSQ, *LCLQ...
User names . . . . . > QMUSER       Name, *PUBLIC
      + for more values
Authority . . . . . > *CONNECT      *ALTUSR, *BROWSE, *CONNECT...
      + for more values
Message Queue Manager name . . . MQ400
-----

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 9-4 Queue manager connect authorization

```

Grant MQ Object Authority (GRMQMAUT)

Type choices, press Enter.

Object name . . . . . > PORTFOLIO.BATCH.COMPLETE
-----
Object type . . . . . > *Q           *ALL, *Q, *ALSQ, *LCLQ...
User names . . . . . > QMUSER       Name, *PUBLIC
      + for more values
Authority . . . . . > *PUT           *ALTUSR, *BROWSE, *CONNECT...
      + for more values
Message Queue Manager name . . . MQ400
-----

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 9-5 Queue put authorization

MQ object authority is granted.


```

Grant MQ Object Authority (GRIMQMAUT)

Type choices, press Enter.

Object name . . . . . > PORTFOLIO.BATCH
-----
Object type . . . . . > *Q          *ALL, *Q, *ALSQ, *LCLQ...
User names . . . . . > QMUSER      Name, *PUBLIC
      + for more values
Authority . . . . . > *GET          *ALTUSR, *BROWSE, *CONNECT...
      + for more values
Message Queue Manager name . . . MQ400
-----

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 9-6 Queue get authorization

Now you can log on as BOB and test using CALL PGM(QMQM/AMQSPUT0) PARM(PORTFOLIO.BATCH.COMPLETE MQ400) at the command line.

```

Call Program (CALL)

Type choices, press Enter.

Program . . . . . > AMQSPUT0      Name
Library . . . . . > QMQM          Name, *LIBL, *CURLIB
Parameters . . . . . PORTFOLIO.BATCH.COMPLETE
-----
      + for more values  MQ400
-----

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 9-7 Call sample put

Sample programs are available, as on other platforms with similar usage. No errors should be returned.

```

Sample AMQSPUT0 start
target queue is PORTFOLIO.BATCH.COMPLETE
>
Sample AMQSPUT0 end
Press ENTER to end terminal session.

===> █

```

```

F3=Exit F4=End of File F6=Print F9=Retrieve F17=Top
F18=Bottom F19=Left F20=Right F21=User Window

```

Figure 9-8 Put message

Also test the get CALL PGM(QMQM/AMQSGET4) PARM(PORTFOLIO.BATCH MQ400).

```

Call Program (CALL)

Type choices, press Enter.

Program . . . . . > AMQSGET4█ Name
Library . . . . . > QMQM Name, *LIBL, *CURLIB
Parameters . . . . . > PORTFOLIO.BATCH

```

```

+ for more values > MQ400

```

```

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display Bottom
F24=More keys

```

Figure 9-9 Call sample get

9.4.2 Further reference

The manuals listed below were used for the setup illustrated above and can be referenced for further detail.

- ▶ *Using MQ on AS/400*, SG24-5236-00
- ▶ *iSeries System Administration Guide*, SC27-1136-00



Architectural vulnerabilities

Using SSL to provide link level security between Message Channel Agents (MCAs) assures customers that malicious attackers cannot read or tamper with in flight messages; it also allows queue managers to authenticate one another.

There are, however, additional security concerns when WebSphere MQ is used in an enterprise environment which SSL does not address. There are two such concerns that we have chosen to discuss in this redbook: starting CICS tasks under the authority of the CICS trigger monitor, and queue manager aliasing.

10.1 CICS trigger monitor

The IBM supplied WebSphere MQ Trigger Monitor (CKTI) runs under the authority from which it was started. If it was started using the default PLTPI mechanism, the user ID field associated with the task will be set to PLTPUSER, which means that for all incoming messages, the started tasks (the ones that process the data) will all be associated with this user identifier. If the CKTI transaction was started by a user from a CECI command for example, then started tasks will run under the authority of the user issuing the CECI command.

Most customers do not wish all tasks to run under a single authority. In an ideal environment, a single task invocation would be associated with the authority of the individual message that is being processed.

10.1.1 Problem solution

There are three solutions to this problem. The first is to obtain supervisor state and modify the user identifier in the task's Task Control Area (TCA). This is not a viable option.

The second is to replace the IBM supplied Trigger Monitor program with a user written one that performs additional logic allowing it to associate the user ID contained in the message descriptor with the started task. While effective, many customers do not feel comfortable with this option since triggering is the main mechanism for invoking programs that issue WebSphere MQ verbs.

The third option is to maintain the CKTI but to introduce a new application layer between this IBM program and the task responsible for interfacing with customer transactions. This is the solution that we have elected to implement for our business scenario. A Cobol solution was used in our business case scenario and the source code has been included in Appendix B on page 265 for interested readers.

Processing logic

The CKTI will monitor an initiation queue (bearing in mind that one can have multiple CKTI tasks running against multiple initiation queues). This is a long running CICS task usually started at PLTPI time by CICS itself.

When a "GET with WAIT" on an initiation queue is satisfied from the trigger monitor (through the queue manager which notices that a trigger condition has become valid) and a message is PUT into the named initiation queue, the trigger monitor simply issues an **EXEC CICS START** command, passing the data it found in the initiation queue message as DATA on the start command (this structure is

named the MQTM). It is at this point that the new thin layer needed to change the user ID associated with the task is invoked.

Instead of the trigger monitor starting transaction ABCD as would normally be the case, it will start transaction EFGH, a newly named transaction. The program that this transaction points to then performs the following logic path:

```
RETRIEVE MQTM (Passed to it by CKTI)
  OPEN INPUT QUEUE NAMED IN MQTM FOR BROWSE
  GET FIRST MESSAGE
  IF BACKOUT COUNT GREATER THAN ZERO
  THEN MESSAGE HAS BEEN ROLLED BACK BEFORE (POTENTIALLY A POISON MESSAGE,
  SO)
    INQUIRE ON QUEUE TO GET BACKOUT THRESHOLD AND
    BACKOUT QUEUE NAME.
    IF BACKOUT THRESHOLD EXCEEDED THEN
  WRITE MESSAGE TO BACKOUT QUEUE ( MQPUT1 WITH AN OPTIONAL DLQ HEADER)
    DESTRUCTIVELY GET THE POISON MESSAGE
    SYNCPOINT
  END IF.
  ELSE CONTINUE
  END IF.
  VALIDATE USER ID CONTAINED IN MQMD (Optional)
  CHECK THAT THE "REAL" TRANSACTION EXISTS AND IS ENABLED - name in
  TRIGDATA for example EFGH
  EXEC CICS START THE "REAL" TRANSACTION WITH USER ID OBTAINED FROM MQMD.
  EXEC CICS RETURN.
```

Note that this new task must execute under a user ID defined to the ESM (RACF, ACF/2, Top Secret) as a surrogate user for the user ID that appears in the MQMD. A surrogate user is a user ID authorized to start tasks on behalf of another user.

It is interesting to note that the functions performed by this new program will closely map to the functions performed by the trigger monitor itself; the only extra thing that the IBM program does is to perform a GET from the initiation queue.

This means that as customers become more comfortable with WebSphere MQ in an operational environment, this program could easily lend itself to replacing the supplied trigger monitor. It should therefore be written with care, and with full compliance with good programming practices (such as setting FAIL_IF QUIESCING) as outlined in the IBM WebSphere MQ documentation.

10.1.2 Additional logic

Some customer keep count of how many messages have been written to a backout queue in a given time frame. If this number is excessive then the

standard practice is to issue an **MQSET** to set the trigger to OFF, and issue another alarm stating that the program is shutting down.

If this is done, then the queue must be opened with the **MQOO_INQUIRE** and **MQOO_SET** options along with current open options.

Another kind of loop that may be encountered is when the triggered program abends or quits after **MQOPEN** but before an **MQGET**. In this case, the message **BACKOUTCOUNT** never goes up, but the triggered program keeps starting over and over again.

This represents a small window of opportunity that does not occur very often, but it usually indicates a rather severe problem that is not transient. In other words, it probably will not go away without manual intervention of some sort. Nevertheless, if you want to program defensively, consider one or both of the following:

- ▶ Use **MQSET** to disable triggering immediately after the **MQOPEN** and then reinstate it after the first **MQGET**
- ▶ Use **MQSET** to disable triggering just before **MQCLOSE** if no **MQGETs** were completed

In either case, if triggering is disabled at **MQCLOSE**, there is a problem that needs attention and you should consider issuing an alert.

You should also be aware that if the triggered program aborts/quits before the **MQOPEN**, triggering will effectively shut down. You can avoid that to a degree by setting a reasonable **TRIGINT**, a queue manager parameter that indicates the default trigger interval and a time at which triggers are reset automatically. However, this tends to mask an underlying problem that needs attention. Therefore, you may want to monitor or use threshold events on the queue depth of the triggered queue.

10.2 Queue manager aliasing

In our business scenario, we had a business-to-business WebSphere MQ link between our z/OS sysplex and an AS/400.

Because we were working in a complex environment with WebSphere MQ clusters, we were concerned that messages could arrive into our environment destined for queues to which the business partner was not allowed to put messages. We were concerned about the admin queues and about the transmission queues which the clustering technology used to pass messages through to other queue managers.

The solution that we developed for this potential problem was to write a channel exit program that interrogates all messages flowing into a receiver channel, and only allow messages through that were destined for named local queues. We created a “good” list of queues in a namelist structure and we placed all other traffic in new queues or in the dead letter queue.

This meant, for example, that if our agreement with our trading partner allowed them to write to a queue named AS400.REPLY then that queue name would be placed in the namelist. If any other messages arrived for a different queue that was not in the namelist, then the queue name would have .SECURE appended to it and the data would be written to that queue, and if the .SECURE version of the queue did not exist, then the message would be written to the dead letter queue.

The intent of this mechanism was to assure our auditors that our trading partners could only write to pre-agreed queues and that they could not make use of services such as remote administration, which would compromise the integrity of our environment.

In our business scenario, we trusted the partner bank but not the WebSphere MQ connections inside the partner bank which might enable others to use that bank as a gateway into our bank. Our intent was to “trust, but verify”.

The source code is included for this receiver message exit in Appendix B on page 265 of this redbook. This example code generates a log file named clamp.log to which it writes extensive messages. The code was written to demonstrate the concept and it is not intended to function as a production quality implementation since it has performance constraints that would negatively affect many customers.

10.3 What SSL does and does not provide

Let’s look at what SSL in WebSphere MQ V5.3 *does* provide. It provides a way of authenticating queue manager to queue manager and WebSphere MQ Client to queue manager connections. A certificate is given to a queue manager and then it is proven that it uses the very same certificate when it starts the channel. Provided that nobody has stolen a copy of the certificate, mutual authentication is performed.

After this reasonably quick and a very friendly handshake has taken place, all future message exchanges are encrypted using an algorithm that was named in the cipherspec exchanged at channel startup time. The choices for creating a secret key used to encrypt these messages are, according to the table shown

below (taken from the MQ 5.3 Security Manual (SC34-6079-01), RC2, RC4, DES, TDES or AES.

Table 10-1 CipherSpecs that can be used with WebSphere MQ SSL support

CipherSpec Name	Hash Algorithm	Encryption Algorithm	Encryption Bits
NULL_MD5 ¹	MD5	None	0
NULL_SHA ¹	SHA	None	0
RC4_MD5_EXPORT ¹	MD5	RC4	40
RC4_MD5_US ²	MD5	RC4	128
RC4_SHA_US ²	SHA	RC4	128
RC2_MD5_EXPORT ¹	MD5	RC2	40
DES_SHA_EXPORT ¹	SHA	DES	56
RC4_56_SHA_EXPORT1024 ^{3,4,5}	SHA	RC4	56
DES_SHA_EXPORT1024 ^{3,4,5,6}	SHA	DES	56
TRIPLE_DES_SHA_US ⁴	SHA	3DES	168
TLS_RSA_WITH_AES_128_CBC_SHA ⁷	SHA	AES	128
TLS_RSA_WITH_AES_256_CBC_SHA ⁷	SHA	AES	256
AES_SHA_US ⁸	SHA	AES	128
Notes@: 1. On OS/400, available when either AC2 or AC3 are installed. 2. On OS/400, available only when AC3 is installed. 3. Not available for z/OS. 4. Not available for OS/400. 5. Specifies a 1024-bit handshake key size. 6. Not available for Windows. 7. Available for AIX, HP-UX, and Linux for Intel platforms only. 8. Available for OS/400, AC3 only.			

The Hash is only used at channel startup time, so let us concentrate on the secret (or *symmetric*) key used by the encryption algorithm.

The secret key is held by both ends of the channel (the Message Channel Agents). These are identical secret keys that were freshly created for them and were exchanged during the SSL handshake.

Keys are only as good as their length; remember that the number of possible combinations doubles for every bit that is added to a key, so a four bit key (0000 to 1111) has sixteen possible combinations, whereas a three bit key (000 to 111) has only eight possible combinations (half). If you look back at the WebSphere MQ CipherSpec chart, you can see that you can have key sizes of 40 bits (RC2 and RC4), 56 bits (DES and 56 bit RC4), 112 bits (T-DES) 128 bits (128 bit RC4 and AES) and 256 bit (AES).

10.3.1 Why key length is important

In January 1996, a distinguished group of Cryptographers and Computer Scientists (Blaze, Diffie, Rivest, Schneier, Shimomura, Thompson and Wiener) published a seminal work entitled *Minimal Key Lengths for Symmetric Ciphers to provide Adequate Commercial Security*. See for details:

<http://theory.lcs.mit.edu/~rivest/bsa-final-report.txt>

They produced a chart that showed how long it would take to attack with brute force 40 and 56 bit keys (start at 1, then try 2, then 3 and so on) for a given amount of money.

They took into account Moore's Law (computer power doubles every 18 months) and in late 1995, they estimated that if you could "borrow" the computing power of a corporate department, you could find a 40 bit key in about 24 seconds and a 56 bit key in about 19 days.

Since the time that the chart was created, there have been four of Moore's 18 month cycles. So the speed of an attack, given the same "borrowing" of mainframe computing power, would be about 3 seconds for a 40 byte key and about 1 and a half days for a 56 bit key.

10.3.2 Why is this important?

This is important because of key re-use. WebSphere MQ uses the same symmetric key it created at channel startup for all message transfers on that channel. It uses the same key over and over again until the channel stops, which will result in a new symmetric key being created when the channel restarts.

Most customers have long running channels, which that stay alive forever or until the queue manager is shut down. Look at DISCINT in your current channel definitions; most of them will be set to DISCINT(0).

Let us say that someone plugged a laptop into your network, using any ethernet port (as is true for most Wireless base stations, allowing this to be done from outside a building), and started "sniffing" data packets using one of the hundreds of available tools. Given that most attacks emanate from inside the organization

and assuming that the TCP/IP address and port to listen to has been located, let us suppose that the sniffer caught a packet sent between two queue managers and found an encrypted message using a symmetric key which was a 56 bit key. As per the chart above, it will take a day and a half on the mainframe or a bit longer on a 3GHz Thinkpad (perhaps a week or two), because the channel never ends, for the program to find the text MQMD in a message and get the key.

The sniffer may not stop after capturing the first data packet but continue capturing all of them and therefore, may catch not just one message but all messages from channel startup to channel shutdown.

One message on its own might be of no interest, but with tens of thousands of messages, there is usually treasure to be found.

Let's suppose that the secure SSL link protected fund transfers between different accounts in a bank. Just after the sniffer plugged into the LAN, two accounts, A and B, were opened. The money was put into one of them and then it was transferred from one account to the other. There would now be some plain text far more interesting to search for than just MQMD. The search is for account numbers; a week or two later, additional funds have been transferred out of an account along with the other funds.

WebSphere MQ messages are atomic, they stand alone. All a sniffer needs to do is to construct the perfect message and introduce it into the network before the channel encrypts it, or create the perfect funds transfer message, encrypt it using the secret key and introduce the packet into the network. This would cause the channel sequence numbers to fall out of sync, but this almost always results in administrators resetting them without looking too closely for an underlying reason; meanwhile, the money would be long gone in our example scenario.

10.3.3 40 and 56 bit key summary

Using a CipherSpec that uses 40 bit keys provides a level of security that can be compromised in a very short period of time, so we do not recommend using it in a commercial enterprise.

Using a 56 bit (DES) key over and over again should only be done where the group of messages sent between channel startup and channel shutdown will prove of no value to someone who figures out the key. For high value transactions, this option is unsuitable.

10.3.4 Using a key larger than 56 bits

Consider the WebSphere MQ chart of supported CipherSpecs and the US government's rules for export of strong cryptography. If you look carefully at the

CipherSpec chart, you will see that not all symmetric key ciphers work on all platforms. Therefore, think about your platforms and then find the lowest common denominator with a key length larger than 56 bits; these are RC4, TripleDES and AES. A brief description of all three follows.

RC4

RC4 was developed in 1987 by Ron Rivest (the R in RSA). It has been quietly used in a variety of applications since then but came to prominence when it was adopted as the security foundation for the 802.11b WEP (Wireless Equivalent Privacy) wireless LAN standard.

A growing number of published studies have found significant weaknesses in the structure and key generation of RC4, prompting the claim by a number of commentators that the algorithm is “unsafe at any key size” (Source: *SearchSecurity.com*, June 2002).

DES

DES was introduced in the early 1970s as the American standard encryption algorithm. It has been widely adopted. There are no viable shortcuts for discovering which key was used in some of the given encrypted data. The only practical approach is the “brute force” method of trying all possible keys in turn until the one that was used is found.

Triple-DES (3DES)

Triple-DES is a much stronger algorithm than DES. It uses a double length key (112 bits) and performs three DES operations, one after the other.

Triple-DES is 256 times stronger than DES. If you could crack a DES key by brute force in 1 second, it would still take two billion years to crack a Triple-DES key in the same way.

AES

AES is the Rijndael (pronounced Rhine Doll) encryption algorithm chosen in October 2000 by the NIST as the new Advanced Encryption Standard to officially replace DES.

AES has key sizes of 128, 192 and 256 bits. To give you some idea of how secure AES is believed to be, here's how the NIST compares it to DES:

“Assuming that one could build a machine that could recover a DES key in a second (to try 2 to the power of 56 keys per second), then it would take that machine approximately 149 thousand-billion (149 trillion) years to crack a 128-bit AES key. To put that into perspective, the universe is believed to be less than 20 billion years old”.

10.3.5 CipherSpec recommendation

Our strong recommendation for CipherSpec for WebSphere MQ is to use a symmetric key length of at least 112 bits, and to regularly stop and restart channels in order to force the exchange of new symmetric keys. The choice of a hash algorithm is at the discretion of the customer. MD5 and SHA-1 are supported hash algorithms and both considered to be strong.

10.4 Attack types

The Computer Security Institute and FBI crime and security survey on security attacks are published at the following Web site:

<http://www.gocsi.com>

The attack types listed are: unauthorized insider accesses, theft of proprietary information, telecom fraud, financial fraud, sabotage, system access by outsiders and intrusion from the Internet and intranet.

The attack trees discussed in Chapter 2, “Planning” on page 9 of this redbook clearly show that the greatest risk of attack for both financial gain and “revenge” (the two primary attacks that concern us) come from unauthorized insiders, a fact confirmed by the above named study. This section of the redbook will focus on software practices and procedures that can be implemented to mitigate these vulnerabilities.

10.4.1 Denial of service

In the WebSphere MQ context, denial of service really means having one or more queue managers made to be unavailable. This can be because the queue manager is overwhelmed with work (this is similar to flooding a Web site with page requests from non-existent TCP/IP addresses), or because it or one of its objects (such as a channel) is placed in an unexpected state.

Being overwhelmed with work is the easier of the two threats to deal with. Remembering the motto stated at the beginning of this redbook: prevention, detection and reaction.

A flooding attack directed at WebSphere MQ can be started accidentally (for example by a program in an unintended send message loop) or by a deliberate act. The unintended attack is prevented by good practices for moving programs through development, quality assurance and then the production cycle.

The deliberate act may be initiated by someone inside or outside the enterprise. Such attacks may be designed to paralyze an organization, for example,

preventing them from executing trades with attached financial risk such that they cannot be performed in a timely manner. Or the attacks may be for financial gain achieved by stock price manipulation, or simply for revenge.

Trying to prevent a WebSphere MQ denial of service attack is almost impossible to achieve. Anyone, for example, can easily send TCP/IP packets to a listener that will overrun that listener. You can mitigate the effect of such attacks by having a dead letter queue defined, implementing clustering as described earlier in this redbook and implementing queue sharing groups on z/OS (if applicable). You can also mitigate the effects of such attacks by implementing multiple channel paths between queue managers.

Rapid detection of such an attack is achieved through WebSphere MQ instrumentation and through third party monitoring products. The QHIGH instrumentation event can be set on queue objects. When the QHIGH threshold is reached, the queue manager writes a message to the SYSTEM.ADMIN.QUEUE which can then trigger an application program that reacts to the situation. The most likely candidates for attack are the dead letter queue, supplied queue objects such as the SYSTEM queues and channels, mainly because they require the least expertise to attack.

The task of a triggered application or situation created by a third party monitor is to stop the attack or, if it cannot be stopped, to reroute the traffic away from the affected component. In the case of a dead letter queue or a SYSTEM queue, the application should alert the middleware administrators to the event; it should then remove messages from the queue and spill them to disk files or to removable media. Autonomic actions such as this are designed to provide a period of time during which system administrators can diagnose and repair the situation.

In the case of an attack on a channel, the autonomic reaction should be to stop the channel, to possibly stop the listener and restart it on a different TCP/IP address using facilities such as VIPA (Virtual IP address) on z/OS.

10.4.2 Fraud

Most WebSphere MQ messages do not contain data of any monetary worth to an attacker. Such messages usually need little or no protection.

Messages that do have monetary worth need to be protected against tampering; they need to be prevented from being replayed, and must be made invisible to observers who may derive worth from being able to see the data. This is where SSL support in WebSphere MQ can be deployed to supply all three of these needs while messages are in flight between queue managers.

However, SSL provides no protection for messages sitting unprotected and unencrypted on local message queues, on transmission queues and in files that will be turned into queues.

In order to protect messages in these repositories, application to application security needs to be deployed (using products such as the Tivoli Access Manager for Business Integration or MQSecure by Candle Corporation). Failing that, these repositories need to be protected using the facilities provided by the operating system. This means Access Control Lists (ACLs) on UNIX platforms, an External Security Manager (ESM) on z/OS (such as RACF) and permissions on Windows platforms.

10.4.3 Password attacks

Contrary to popular belief, passwords are not stored in files by software such as WebSphere MQ. What is stored is a digital hash of the password. This is why systems administrators can always reset passwords, but can never tell you what your password is: nobody knows. All that the software knows is how to turn what you type into a hash and then compare that hash with the stored hash. If they match, you typed the correct password or managed to type something that caused a "collision" that computed an identical hash for a different value (which can happen every billion years or so).

Nevertheless, the files or devices used to store these hashes are extremely valuable and need strong protection. This is because it is much easier to attack a short password than it is to attack a strong crypto key. Remember, a five digit password is $5^8 = 40$ bits long and the earlier chart shows that a brute force/dictionary attack on 40 digits takes almost no time at all. The popular LC4 password cracker for Windows in the example below took less than half a second to find the five digit password "peter". For reference, see the following link:

<http://www.atstake.com/research/lc/>

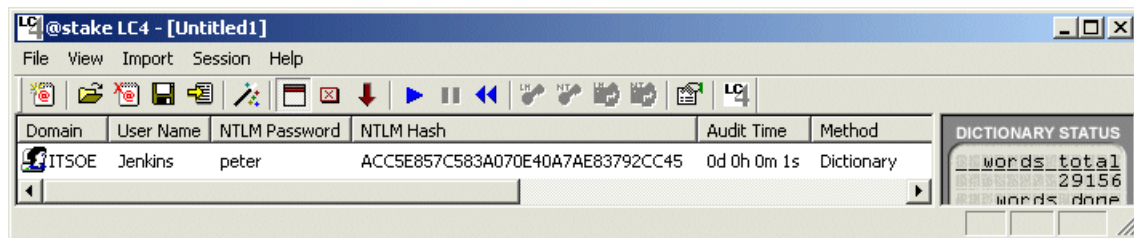


Figure 10-1 Example of password cracker

An easy password attack in a z/OS environment is to know the user ID of the RACF administrator, start a TN3270 session, type the user ID and press **Enter**

three times. Then phone the help desk and say that RACF revoked your ID (which it will have done) and ask for it to be reset. Most help desk staff will reset the password straight away and some will automate the process and send you an e-mail. This can, of course, be intercepted and read and if it is encrypted, remember that Notes uses a 40 bit key to encrypt an e-mail.

Another attack would be to install a keystroke capture program (many of which are widely available on the Web) on one of the machines that belongs to a WebSphere MQ administration staff.

Detecting and deleting keystroke capture programs is difficult. A good procedural recommendation is to compare registry and startup options on Windows machines with known “good” images, and quarantine the machine if differences are found.

Systems administrators using z/OS should note that RACF can only accept eight digit passwords which yield a key length of 64 bits. According to the experts, this will take several years of brute force attack, although an attack on the dictionary might find the password earlier.

Creating strong passwords for anything remotely associated with WebSphere MQ is a good idea. An easy way to do this is to use the iKeyMan utility that ships with WebSphere Application Server HTTP Server. This can be downloaded for free (see Chapter 1, “Project overview” on page 3) and gives you a visual representation of the password strength. Even with an eight digit password in z/OS, with iKeyman, it is possible to create what is considered to be a “strong” password.

10.4.4 Access to certificates

Certificates that contain private keys imbedded within them are in PKCS#12 format or have been imported into a repository with exportable private keys. These are extremely dangerous entities because, if you know or can guess the password protecting the PKCS#12 file (which is embedded inside the file in hash format), then you can clone duplicates and compromise the integrity of the certificate. Procedurally, strong pass phrases must be used to protect PKCS#12 files.

10.4.5 Access to files

Certificates, queues, keyrings, object definitions and other WebSphere MQ elements are all stored in files. Adequate security exists in almost all operating systems that allows for read, write, execute and delete type access control for files and should be implemented for all WebSphere file locations. Not protecting such files renders technology such as SSL meaningless.

10.4.6 Access to binaries (executables)

Access to the binaries that make up the WebSphere MQ product set must be controlled strictly in any production environment. The executables need to be secured and protected and unauthorized modified versions must be prevented from being introduced into a production environment.

10.4.7 Access to WebSphere MQ object definitions

Websphere objects can be manipulated using several interfaces such as the **RUNMQSC** command line utility, the MQ Explorer interface and several third party products that support the Programmable Command Facility (PCF) syntax. The users of these products need to be identified and specifically granted access through the OAM and the ESM on z/OS.

10.4.8 LDAP access

LDAP is used by WebSphere to manage Certificate Revocation Lists (CRLs). Obviously, if CRLs are used by the enterprise, they need to be secured against fraud (removing a revoked certificate from a store) and denial of service (adding valid certificates to the CRL). LDAP supports a user ID and password passed through the LDAP API calls. As with all other passwords discussed in this chapter, this password needs to be strong and changed at regular intervals. The LDAP data repository should be secured using platform security, and definitions holding the TCP/IP address and port used by the LDAP server need to be carefully protected against subversion.

10.4.9 Wireless LANs

Wireless Access Points using the popular 802.11b (WiFi) standard comes with a default SSID (an identifier that is broadcast over the airwaves) and with all security set to OFF as the default. This means that you can stand outside most large office buildings with a Laptop or a PocketPC and use a popular tool such as NetStumbler (<http://www.netstumbler.com>) to find an access point (usually called Linksys or Airport). Assuming that the security for the SSID is turned off, which Netstumbler shows with a padlock icon, then the access point name is entered. In almost all environments, the access point will route traffic to a DHCP server that will provide the machine with an TCP/IP address, a gateway address and of course the DNS suffix of the parent company.

A simple port scan looking for well-known WebSphere MQ ports 1414, 1415 and so on, will then usually yield a number of queue manager listeners, which when pinged will return a DNS name that is often the same name as the queue manager's. It is then a simple task to right-click **WebSphere MQ Explorer** and

add the queue manager to its window, manipulate objects and find routes to other clusters and queue managers. In a demonstration from a parking lot, one of our team was able to use these techniques to prove that we were capable of sending command messages to a production z/OS queue manager through its SYSTEM.COMMAND.INPUT.QUEUE by implementing a few simple remote queue definitions that exploited queue manager aliasing. Clearly, this is unacceptable in all organizations.

The risks associated with allowing unencrypted Wireless Access Points to be created inside an enterprise cannot be overemphasized. Such devices compromise the integrity of the entire enterprise, since they essentially allow free access to all systems located inside corporate firewalls. Building SSL connections between queue managers does nothing to stop this problem. The risk can only be mitigated from a WebSphere MQ standpoint through use of Access Control Lists, MCAUSER use, and use of all of the other techniques detailed in this chapter. Procedures to detect and disable unencrypted Wireless Access Points need to be implemented and constantly monitored by all enterprises that value their WebSphere MQ environment.

10.5 Cryptographic co-processors

WebSphere MQ supports cryptographic processors where these are available through standard operating system interfaces. For this redbook, we used Triple DES as our symmetrical key algorithm. Unfortunately, the hardware cryptographic boards installed in our Sysplex were incapable of supporting triple DES, AES or RC4, our preferred symmetric key algorithms.

We therefore elected not to use the cryptographic co-processor. We did, however, configure the device before determining that it could not support our requirements. The following sections illustrate the configuration.

Before you can use the cryptographic co-processor in a z/OS environment, the device must be enabled in microcode, and the Integrated Cryptographic Support Facility (ICSF) must be configured and the started task run. The following JCL section illustrates the parameters used by our started task.

```
//CSF PROC SUFFIX=00
//EXEC PGM=CSFMMAIN,REGION=6M,TIME=1440
//CSFLIST DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=132),HOLD=YES
//CSFPARM DD DSN=SYS1.PARMLIB(CSFPRM&SUFFIX),DISP=SHR

CKDSN(WTSCPLX1.SCSFCKDS)
PKDSN(WTSCPLX1.SCSFPKDS)
PKDSCACHE(64)
COMPAT(NO)
```

```
SSM(NO)
KEYAUTH(YES)
CHECKAUTH(YES)
TRACEENTRY(1000)
USERPARM(USERPARM)
COMPENC(DES)
REASONCODES(ICSF)
DOMAIN(4)
```

Startup messages are written to the z/OS console.

```
000290 S CSF,SUFFIX=62
000090 $HASP100 CSF      ON STCINRDR
000290 IEF695I START CSF      WITH JOBNAME CSF      IS ASSIGNED TO USER STC
      , GROUP SYS1
000090 $HASP373 CSF      STARTED
000010 IEF403I CSF - STARTED - TIME=23.27.20 - ASID=03EA - SC62
000090 IEE504I CRYPTO(0),ONLINE
000090 IEE504I CRYPTO(1),ONLINE
000090 CSFM009I NO ACCESS CONTROL AVAILABLE FOR ICSF SERVICES OR KEYS
000090 CSFM405A AUTHENTICATION CODE IN PKDS HEADER RECORD DOES NOT MATCH
      COMPUTED VALUE.
000090 CSFM001I ICSF/MVS™  INITIALIZATION COMPLETE
```

Configuration is performed through a series of TSO panels. Shown below is the primary ICSF panel and the values used to configure both of our cryptographic co-processors (they are supplied in pairs for hardware fault tolerance).

```
,----- Integrated Cryptographic Service Facility-----
,OPTION ==>,
,Enter the number of the desired option.
,
, ,1, MASTER KEY - Set or change the system master key
, ,2, KGUP      - Key Generator Utility processes
, ,3, OPSTAT   - Installation options and Hardware status
, ,4, OPKEY    - Operational key direct input
, ,5, UTILITY  - OS/390 ICSF Utilities
, ,6, CKDS     - CKDS Refresh and Initialization
, ,7, USRCNTL - User Control Functions
, ,8, PPINIT   - Pass Phrase Master Key/CKDS Initialization
, ,9, PCICC MGMT - Management of PCI Cryptographic Coprocessors
, ,10, UDX MGMT - Management of User Defined Extensions
,
, ,Licensed Materials - Property of IBM
, ,This product contains "Restricted Materials of IBM"
, ,5647-A01 (C) Copyright IBM Corp. 2000. All rights reserved.
, ,US Government Users Restricted Rights - Use, duplication or
, ,disclosure restricted by GSA ADP Schedule Contract with IBM Corp.,
,
,Press,ENTER,to go to the selected option.
```

```

, F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
  F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE

```

```

,----- OS/390 ICSF - Installation Option Display, Row 1 to 15 of 15,
,COMMAND ==>, ,SCROLL ==>,PAGE,
,      Active CKDS:;WTSCPLX1.SCSFKDS ,
,      Active PKDS:;WTSCPLX1.SCSFKDS ,
, OPTION CURRENT VALUE
, -----
, CHECKAUTH RACF check authorized callers YES
, COMPAT Allow CUSP/PCF compatibility NO
, COMPENC Compatibility services encryption algorithm DES
, DOMAIN Current domain index or usage domain index 4
, KEYAUTH Key Authentication in effect YES
, MAXLEN Maximum data length 65535
, SSM Allow Special Secure Mode NO
, TRACEENTRY Number of trace entries active 1000
, USERPARM User specified parameter data USERPARM
, REASONCODES Source of callable services reason codes ICSF
, WAITLIST Source of CICS Wait List if CICS installed default
,
, PKDSCACHE PKDS Cache size in records 64
,
, Encryption algorithm available DES,CMDF
,***** Bottom of data *****
, F1=HELP      F2=SPLIT      F3=END      F4=RETURN      F5=RFIND      F6=RCHANGE
  F7=UP        F8=DOWN        F9=SWAP     F10=LEFT      F11=RIGHT     F12=RETRIEVE

```




Business scenario solution

This chapter explains in detail what security aims we wished to achieve in our scenario and how we enhanced the security for our business through the deployment of SSL. It explains what was needed on the z/OS, OS/400, IBM AIX and Windows 2000 platforms to implement SSL in our production environment without compromising performance or availability.

11.1 Dolphin Bank International security policy

You can make good decisions about security protection, detection and reaction methods if you have a proper understanding of your security goals. A good policy document should address directly the threats and risks discovered in the threat assessment, vulnerability assessment and risk assessment.

As stated in RFC2196, *“until you determine what your security goals are, you cannot make effective use of any collection of security tools because you simply will not know what to check and what restrictions to impose”*. RFC2196 also defines that *“a security policy is a formal statement of the rules by which people who are given access to technology and information assets in an organization must abide.”*

Our security goal is to further secure WebSphere MQ on all the platforms in our environment. We have defined what we wish to achieve and why we need to achieve it. The policies defined are minimal at present.

11.1.1 Policy for encrypting technology and certificate authority (CA)

The following section discusses security policies and levels of security for the z/OS, OS/400, IBM AIX and Windows 2000 platforms.

Cipher strength

Information about Dolphin Bank International has been categorized into the following areas:

- ▶ Secret
- ▶ Company confidential
- ▶ Company sensitive
- ▶ Restricted
- ▶ Unclassified

The classification distills into the strength of ciphers required in the bank.

Secret information should be enciphered at all times. It should not exist in a clear state at any time. The chosen cipher must be suitable for long term protection (25 years) of the information from cryptanalysis, based on Moore's law.

Company confidential and company sensitive information may exist in the clear when written to a storage medium protected by a secure operating system. Confidential and sensitive data should only be available to defined privileged user roles such as administrators. Company confidential information should be enciphered when it passes on internal and external networks. The chosen cipher should be strong enough to protect the data from cryptanalysis for 25 years.

Restricted data may exist in the clear at all times. It should not be possible to alter, manipulate or replay restricted data when it is moved on internal or external networks.

Unclassified data requires no security. This classification should be restricted to information published in the public domain.

Private key storage

Where data is classified as secret, the private key must be held in a device that is FIPS140-1 Level 2 compliant. Otherwise, private keys may be held on secured hard drives protected by the operating system security. Access to private keys should be limited to specific privileged users and audited.

Certificate Authority

The PKI can be handled in-house. In our environment, the Dolphin Bank International is the CA. The bank can issue certificates to business partners and provide the root certificate for checking. The CA should exist on a secure operating system with the private key held in a secure FIPS140-1 Level 2 compliant device.

Another way of running an internal CA is to get one CA certificate from a trusted third party which acts as the CA root, and then to set up your enterprise as a subsidiary CA to this root.

11.1.2 Policy for WebSphere MQ on z/OS

This section discusses policy and levels of security on z/OS.

Administration security

All security should be set to on at a queue sharing group level, on all the WebSphere MQ for z/OS systems. Nothing should be defaulted to off.

Administration of WebSphere MQ for z/OS should be restricted to a limited number of staff who have the appropriate access to commands on WebSphere MQ for z/OS. Access to the commands should not be given to anyone who does not absolutely need it.

WebSphere MQ for z/OS commands should be controlled using both command security and command resource security. Fully qualified profiles should be defined for all the commands in the MQCMD5 class and fully qualified profiles should be defined in the MQADMIN class for all resources that the commands will act upon.

The use of the operations and control panels should be limited to a few people who need to use them. You should also restrict commands that can be issued from the ops and controls panels by using command and command resource security. This can be done by permitting different levels of access to the limited number of user IDs under which the panels run, and controlling which user IDs are allowed to use which commands. For example, for panels running under user IDs PROD1, DEVT2 and TEST3, you could restrict access by giving PROD1 access only to display commands, DEVT2 access to all commands and TEST3 access to all commands except **DEFINE**, **DELETE**, and **ALTER**.

Connection security

Careful consideration should be given to who you actually want to be able to connect to your queue manager. This should be controlled using the appropriate profiles and accesses in the MQCONN class.

API security

API security should be tightly controlled by ensuring that fully qualified profiles are defined for all queues within the system. Use of generic profiles should be limited to profiles that cover use of dynamic queues. These need a high level qualifier stem in order to be protected because the remainder of the name will be a store clock value.. For example:

```
CSQ.**  
CSQOREXX.**  
CSQUTIL.**  
CSQXCMD.**  
CSQ4SAMP.**
```

Access to the queues should be tightly controlled and only those user IDs that need access to the queues should be given it at the minimum level required.

Alias queues should be used where put only or get only access to a given queue is required. For example, many user IDs may need to have put access to the dead letter queue, but you do not necessarily want them all to be able to get from the queue. An Alias queue should be set up with PUT(ENABLED) and GET(DISABLED); the alias queue could have the stem of its name the same as the real dead letter queue but suffixed with USER or PUT. All applications should then be designed to put to the alias queue and retrieve the real dead letter queue name by inquiring on the queue manager object and appending the agreed suffix.

Context Security should be implemented for all queues; fully qualified profiles should be defined in the MQADMIN class and appropriate access given to the WebSphere MQ CHINIT user IDs, MCA user IDs. General users should be given

the minimum level of access possible to the appropriate context profiles in order for them to perform the required functions.

Alternate user ID security should be implemented where required. Use fully qualified alternate user ID profiles in the MQADMIN class. Use of generic profiles in this area, as in all others, is not recommended.

Channel security

Channel definitions should be set up carefully and with a great deal of consideration as to who will be using them and from where. All channels at the “receiving” end of the link should be set up with the MCA user ID attribute set; it should not be left blank to default to the CHINIT user ID. All channels should be protected using SSL when a secure link is required.

Reslevel security

Reslevel security should be set to enable full resource security checking on all connections to WebSphere MQ. This means that the hlq.RESLEVEL profile should be defined in the MQADMIN class and all connecting user IDs should be granted access(NONE) to this profile.

Message security

If the data content of message is considered a valuable asset or contains private or sensitive information, the messages should be encrypted during transport and when stored, if required.

11.1.3 Policy for WebSphere MQ on OS/400

This section discusses policy and levels of security on OS/400.

Server administration

Administration should be limited to properly trained staff who have sufficient skills and are members of QMQMADM. Administration should only be possible when a user has been properly authenticated to the operating system. Remote administration is permitted, provided the communications are secured from the remote client (not susceptible to spoofing) to the server, and the administrator authenticates to the local operating system. The actions of the administrator need to be audited and accountable. The role of the MQA is global and the MQA has access to all queue managers on a system. The MQA will have access to any cryptographic keys associated with the queue managers.

This policy statement exists to prevent users from completing unauthorized tasks.

Client connections

Client connections are not permitted. Client connections should connect only to the Windows queue managers.

Messages

Information classification determines how messages should be stored and transported. The policy for messages transported and stored should be in line with the bank cipher policy.

11.1.4 Policy for WebSphere MQ on AIX

This section discusses policy and levels of security on AIX.

Server administration

Administration should be limited to properly trained staff who have sufficient skill. The MQA should be a member of the mqm group. Administration should only be possible when a user has been properly authenticated to the operating system. Remote administration is permitted provided the communication is secured from the remote client to the server, and the administrator authenticates to the local operating system. The remote administration should not introduce a weak link. The actions of the administrator need to be audited and accountable. The role of the MQA is global and the MQA has access to all queue managers on a system. The MQA will have access to any cryptographic keys associated with the queue managers.

This policy statement exists to prevent users from completing unauthorized tasks.

Client connections

Client connections are not permitted. Client connections should connect to the Windows queue managers.

Messages

Information classification determines how messages should be stored and transported. The policy for messages transported and stored should be in line with the bank cipher policy.

11.1.5 Policy for WebSphere MQ on Windows

This section discusses policy and levels of security on Windows 2000.

Server administration

Administration should be limited to properly trained staff who have sufficient skill. WebSphere MQ administrators (MQA) should be members of the mqm group. Administration should only be possible when a user has been properly authenticated to the operating system. Remote administration is currently not permitted, and administration can only be performed by a user interactively logging into the server. The actions of the administrator need to be audited and accountable. The role of the MQA is global and the MQA has access to all MQ queue managers on a system. The MQA will have access to any cryptographic keys associated with the queue manager.

This policy statement exists to prevent users from completing unauthorized tasks.

Client connections

Client connections should be validated and authenticated. It should not be possible for them to connect to your queue manager. Access should be limited to resources that have a direct function with the client.

Messages

Information classification determines how messages should be stored and transported. The policy for messages transported and stored should be in line with the bank cipher policy.

11.2 WebSphere MQ objects for SSL

You should set up the queue manager and channels attributes for SSL. You only need to define the SSL channel attributes when your channel exploits the SSL technology.

11.2.1 Queue manager

There are two aspects to setting up a WebSphere MQ queue manager to use SSL. One is to set up the WebSphere MQ queue manager attributes and the other is to set up the WebSphere MQ channel.

Queue manager attributes for SSL

Table 11-1 Qmgr attributes for SSL

Attribute	Description	
SSLKEYR	Key repository name without extension (sto, kdb)	required

Attribute	Description	
SSLCRYP	SSL Cryptographic H/W (UNIX only)	optional
SSLCRLNL	for CRL check	optional
SSLTASKS	Number of SSL tasks (z/OS only)	optional

If you want to set up CRL check, you have to define two MQ objects, NAMELIST and AUTHINFO, specified for the location of the LDAP server.

Channel

Distributed channel attributes for SSL, including cluster channel and SVRCONN channel for client connection.

Table 11-2 Distributed and SVRCONN channel attributes for SSL

Attribute	Description	
SSLCIPH	CipherSpec name (It must be the same as the other channel's)	required
SSLCAUTH	REQUIRED/OPTIONAL for SSL client authentication	optional
SSLPEER	for DN filtering	optional

Following is a diagrammatic representation of SSL definitions.

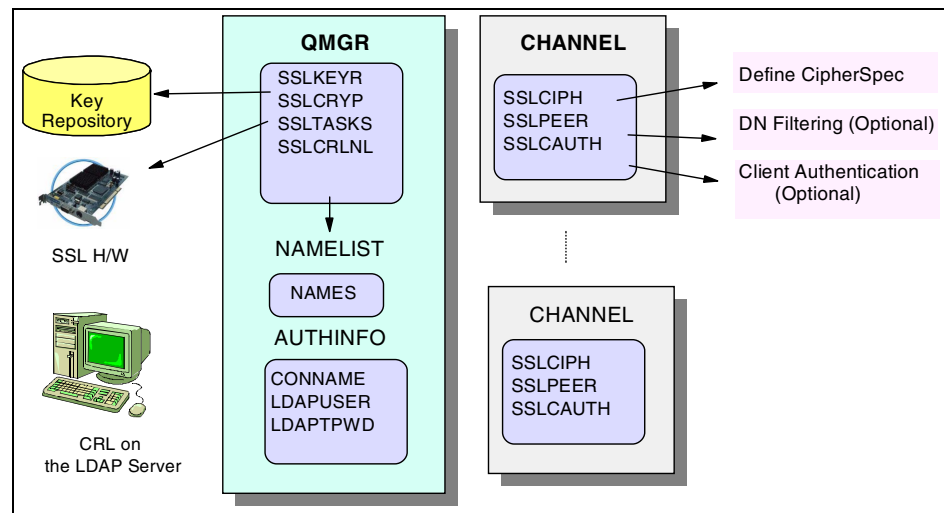


Figure 11-1 SSL setup for queue manager and channel

11.2.2 WebSphere MQ client

There are two aspects to setting up a WebSphere MQ client to use SSL. One is to set up the WebSphere MQ client channel and the other is to set up LDAP CRLs. The latter aspect is not covered in this section, although the figures do include LDAP CRL information. Please refer to the WebSphere MQ manuals for further detail.

WebSphere MQ client channel

There are four methods of WebSphere MQ client channel setting: the MQSERVER environment variable, the WebSphere MQ channel table, Windows Active Directory (Windows only), and MQCONNX API. The MQSERVER variable does not support an SSL connection.

1. WebSphere MQ channel table:

Set up these environment variables for SSL where your client program is running.

Table 11-3 MQ Client environment variables for SSL

Variable	Description	
MQSSLKEYR	Key repository name without extension (.sto,.kdb)	required
MQSSLCRYP	SSL Cryptographic H/W (UNIX only)	optional

Define a CLNTCONN channel, specifying the following attributes on your WebSphere MQ server machine. This creates a WebSphere MQ client table file called AMQCLCHL.TAB, which needs to be copied to your client machine.

Table 11-4 CLNTCONN channel attribute for SSL

Attribute	Description	
SSLCIPH	CipherSpec name (It must be the same as the other channel's)	required
SSLPEER	for DN filtering	optional

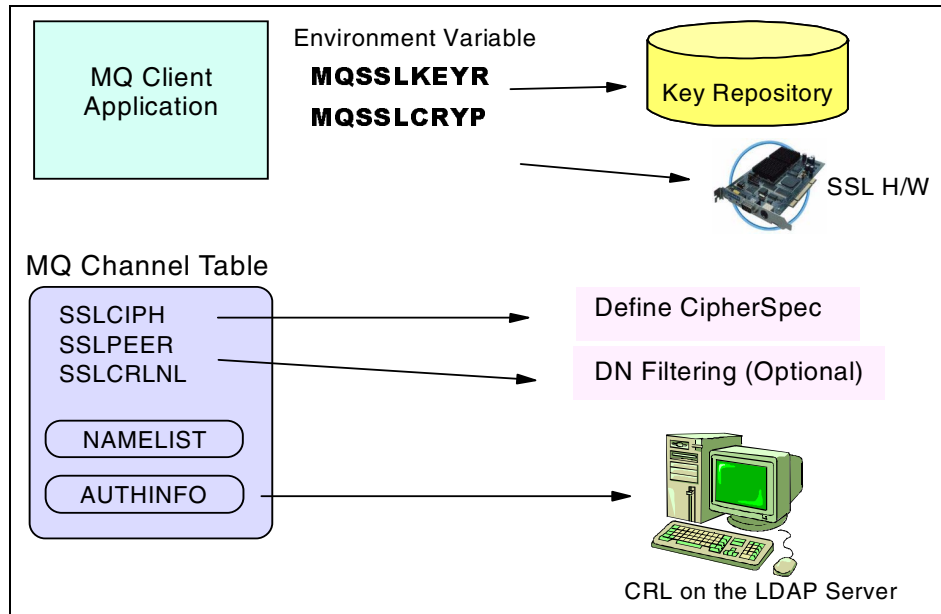


Figure 11-2 SSL setup for WebSphere MQ Client using MQ channel table

2. Windows Active Directory

Instead of using the client channel table to hold the client channel definitions, on Windows these can be held in the Windows Active Directory.

3. MQCONNX in your program

You can specify all the attributes for SSL in your program using the MQCONNX API. This gives you more flexibility and ease of management against the system environment changes than does the WebSphere MQ channel table.

You must use the MQCNO V4 structure specified MQCNO_VERSION_4, the MQCD V7 structure with MQCD_VERSION_7, the new MQCSO structure that is provided for SSL configuration and the MQAIR for the CRL check.

- a. Define a MQSCO address in SSLConfigPtr of MQCNO.
- b. Define SSL CipherSpec. If you use DN filtering, define SSLPeerNamePtr and SSLPeerNameLength.
- c. Define the key repository. If you use cryptographic hardware, define CryptoHardware.

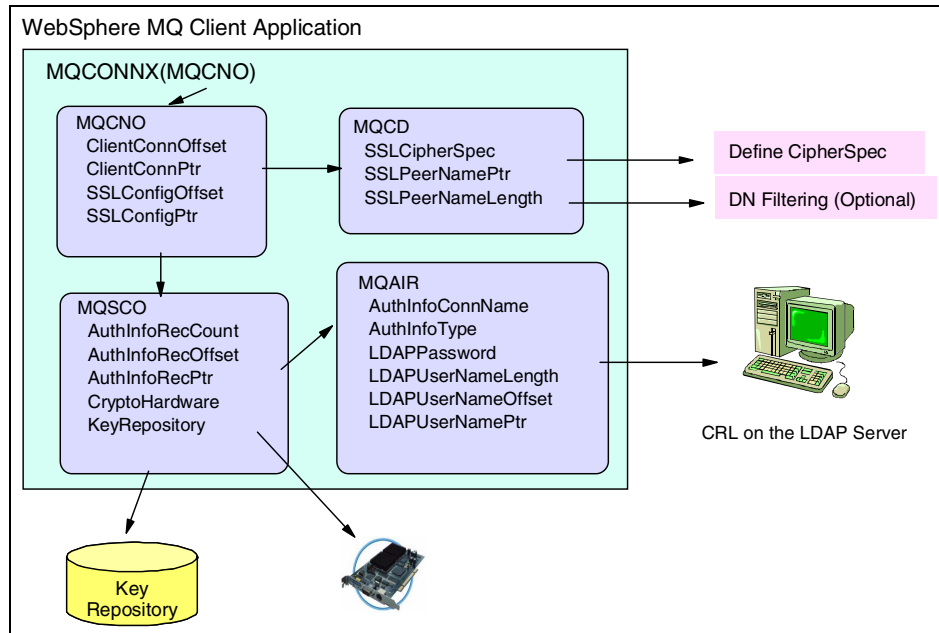


Figure 11-3 SSL setup for WebSphere MQ Client using MQCONNX

11.3 Security setup

The following sections discuss the planning, preparing certificates and WebSphere MQ setup necessary to implement SSL in the production environment at Dolphin Bank International.

11.3.1 Planning

We set up our two secure clusters, called ECLUS1 and ECLUS2, with SSL and a client channel.

In the cluster environment, all the queue managers that wanted to communicate with the CLUSRCVR channel had to have SSL support because the CLUSRCVR channel's attributes were propagated to many other queue managers.

- ▶ The CipherSpecs
TRIPLE_DES_SHA_US

- ▶ The key repository file name:
 - **z/OS:** MQZ1RING, MQZ2RING, MQZ3RING
 - **AIX:** /var/mqm/qmgrs/<QmgrName>/ssl/key
 - **Windows(Server):** <Install directory>\QMGRS\<QmgrName>\SSL\key
 - **Windows(Client):** C:\SSL\mykey
- ▶ Prepare required certificates

We set up RACF on z/OS as the Certificate Authority and used CA signed certificates.
- ▶ SSL Client Authentication

On the cluster channel, SSL Client Authentication was set to on by default, SSLCAUT=(REQUIRED) in the CLUSRCVR definition.
- ▶ DN filtering

Only the organization name *Dolphin* could join the cluster.

O=Dolphin*

CRL check by LDAP was not used.

Only z/OS had cryptographic hardware: the IBM 4758 PCI Card.

11.3.2 Preparing a certificate on AIX

Note: You do not have to compose a certificate request on your own machine. You can do this on one machine and on any platform.

Creating a key repository on AIX

1. Start iKeyman on AIX
 - export DISPLAY=*hostname*:0
 - export JAVA_HOME=/usr/mqm/ssl/jre
 - gsk6ikm& (this takes a few minutes to start)

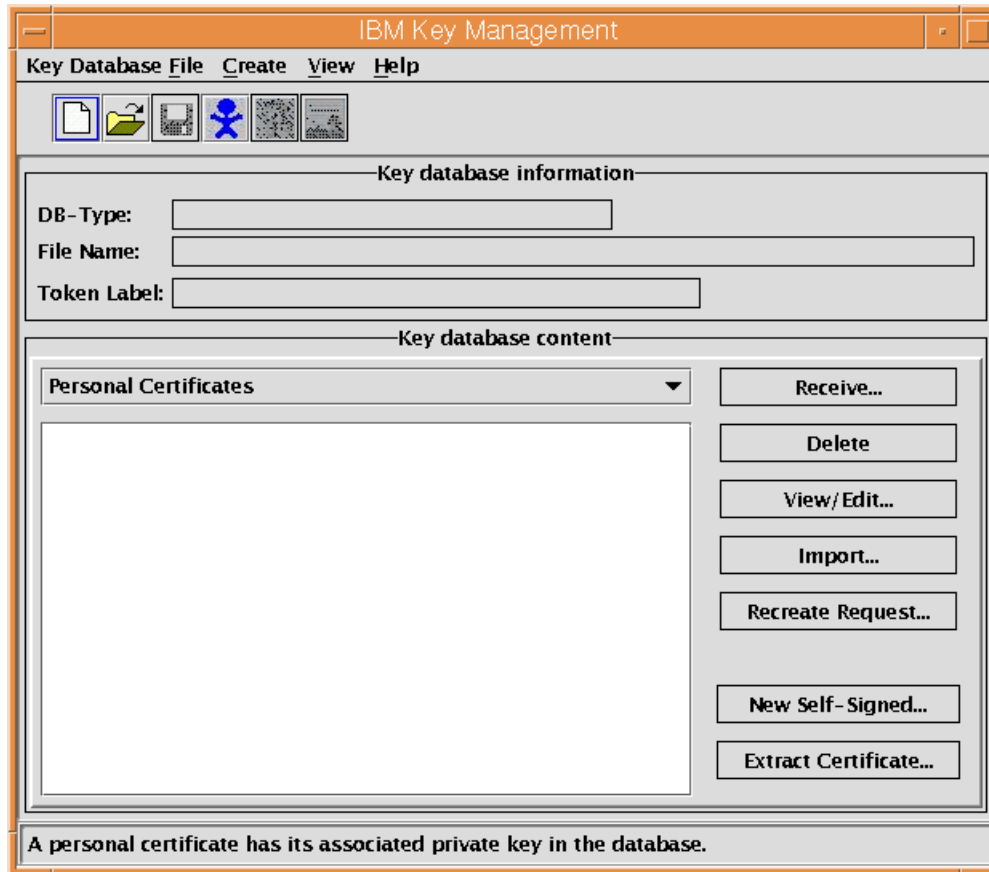


Figure 11-4 Main menu of iKeyman on AIX

2. Create a key repository

Click **Key database File -> New**.

The key repository is called Key database on UNIX.

Select the key database, type CMS and input your file name and location; the file name must have an extension of .kdb.

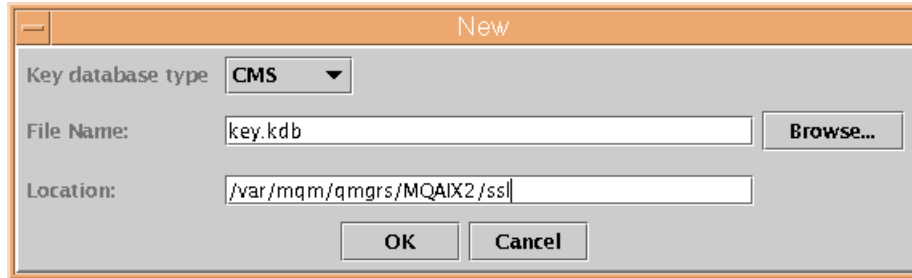


Figure 11-5 Create key repository

3. Set the password.
Select **Stash the password to a file?**.



Figure 11-6 Enter the password of the key repository

The Key database file and password stash file are created in your directory.
For example:

```
/var/mqm/qmgrs/MQAI2/ssl/key.kdb
/var/mqm/qmgrs/MQAI2/ssl/key.sth
```



Figure 11-7 Successful message

Obtaining a CA signed certificate

1. Create a certificate request by iKeyman by clicking **Create -> New Certificate Request**.

Fill in the certificate information.

The key label must be `ibmwebspheremq<QmgrName>` (lower case in AIX).

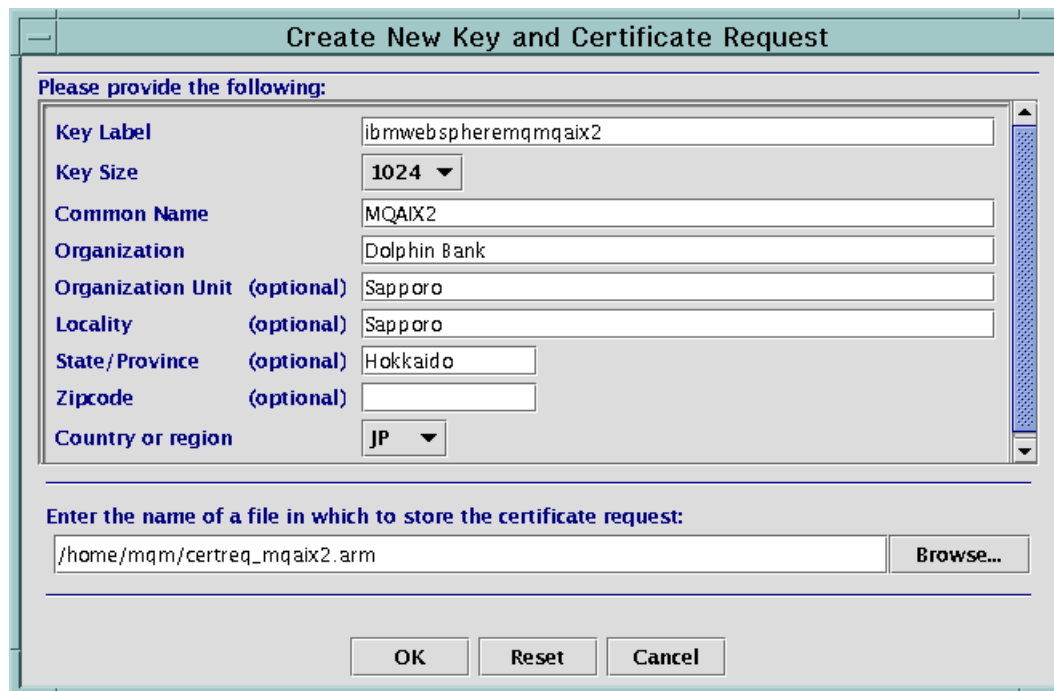


Figure 11-8 Create a certificate request

For example, a certificate request file `/home/mqm/certreq_mqaix2.arm` is created that you then send to your CA and get back as a CA signed certificate.

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrDCCARUCAQAbwDELMAkGA1UEBhMCS1axETAPBgNVBAGTCehva2thaWRvMRawDgYDVQQHEwdT
YXBwb3JvMRUwEwYDVQQKEwxEb2xwaG1uIEJhbmsxEDAOBgNVBAAsTBiNhcHBvcms8XDzANBgNVBAMT
Bk1RQU1YMjCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAj8cqbZ1KbHFdejNp+hev1QdBx1ia
JELCU5xucGMWFzfPT32QC1MPCiJEceOtw+jwg8aRQXaH+VTad2qdMyH8aughBB1PPOeURw982TKF
UAltSuuTtD/aMK9Cm/zibEoFciEMOUcmaZsTSRXZL+Qa1fm/pm471B5C8Le9OCsJdUCAwEAAAaA
MAOGCSqGS1b3DQEBBAUAAA4GBAHKkZm1OT8xfgjph5HvOHwfaeKF/cgyupLrOfTUUQpgZwfwVQDmt
tAVVynOMxBcFvc18tIi/Dgn/GA3QxahcG3s9JQytcCu3OdWokaqgAD4KjWL10iSxr4t5eBCGIjZc
RWEtzMbnzafyWeOR9mcFDiGKrmT1E27XLZAhFfU7MUJT -----END NEW CERTIFICATE REQUEST-----

```

Figure 11-9 Certificate request file (certreq_mqaix2.arm)

Storing your certificate on AIX

1. Add the CA Certificate by clicking **Signer Certificates-> Add...**

Input the CA certificate file name and enter the label of the CA, for example, a file name CERT.CA.der and the label Dolphin Bank CA.

Important: The CA's certificate must be added before your personal certificate.

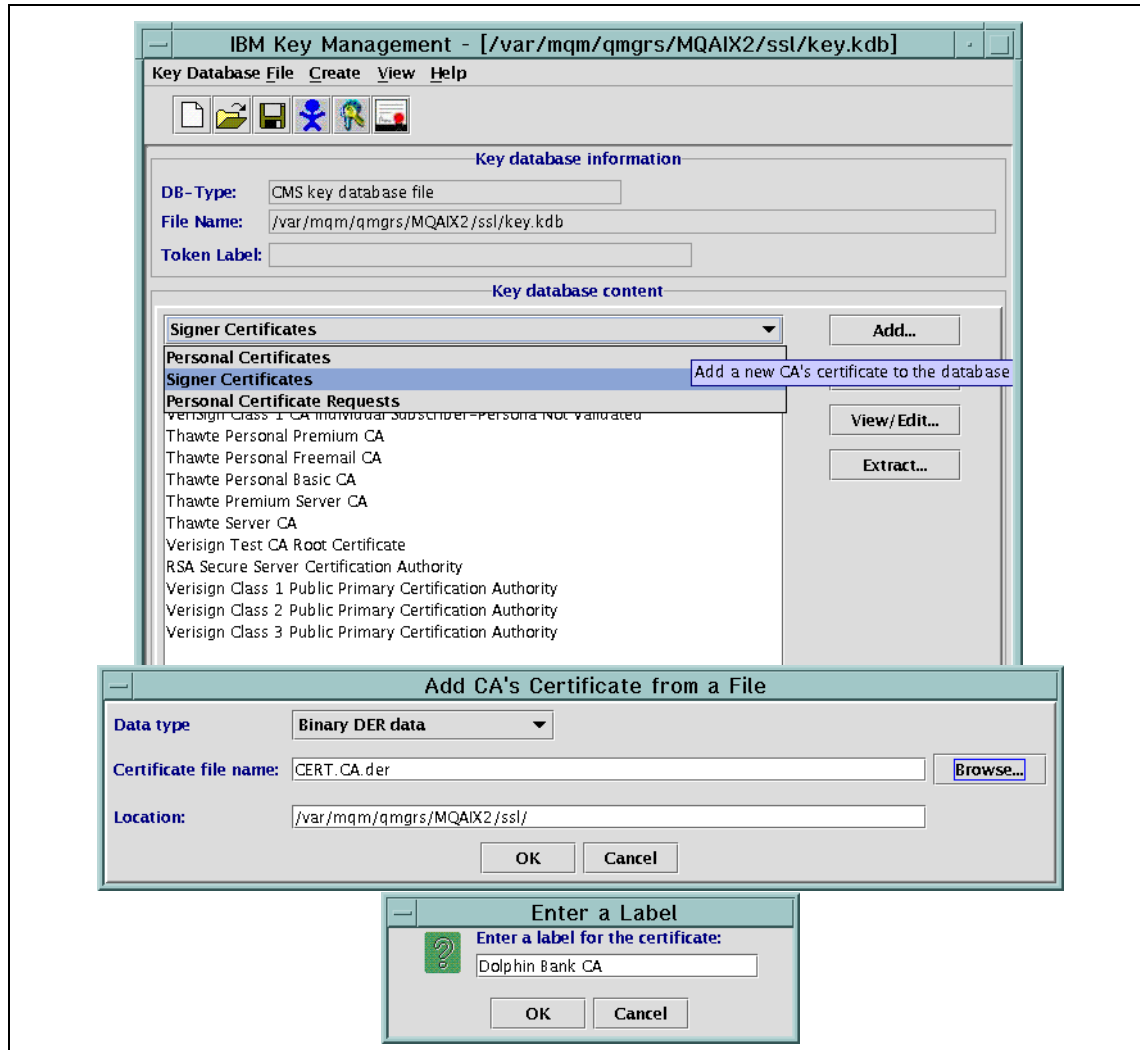


Figure 11-10 Storing the CA's certificate

2. Select **Set the certificate as a trusted root.**



Figure 11-11 Key information of imported CA

3. Receive the CA signed certificate.
Select **Personal Certificates** and click **Receive....**

Important: You must receive your CA signed certificate on the same machine where you created your certificate request.

Input your Digital Certificate file name.
For example, the file name could be CERT.MQAIX2.arm.

```
-----BEGIN CERTIFICATE-----
MIIC4TCCAkqgAwIBAgIBFzANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJVSzEV
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDVQQHEwZUZXRzZXkxEDAOBgNVBAoT
B0RvbHBBoaW4xCzAJBgNVBAsTAkhrMRgwFgYDVQQDEw9Eb2xwaGlueJhbmsgQ0Ew
HhcNMDIxMDMxMDUwMDAwHhcNMDMxMTAxMDQ1OTU5WjBsmQswCQYDVQQGEwJKUDER
MA8GA1UECBMISG9ra2FpZG8xEDAOBgNVBAcTB1NhcHBvcn8xFTATBgNVBAoTDERv
bHBBoaW4gQmFuaZQEQA4GA1UECXMHU2FwcG9ybzEPMA0GA1UEAxMGTUVFBSVgyMIGf
MA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCPyqAFnUpscV16M2n6F6+VB0HHWJok
QsJTnG5wYxYXN89PfZALUw8KIkrx463D6PCDxpFBdof5VNp3ap0zIfxq6CEEGU88
55RHD3zZMoVQCW2xSpS0P9owr0Kb/OJsSgVyIQw5a4KZpmxNJFdkv5BqV+b+mbjv
UHkLwt704Kw11QIDAQABo4GQMIGNMESGCWCGSAGG+EIBDQQ+EzxHZW51cmF0ZWQg
YnkgdGhlIFNlY3VyZVdheSBTZWN1cm10eSBTZXJ2ZXIgzM9yIE9TLzM5MCAoUkFD
RikwHQYDVR0OBByEFGGk3993Pxp+TL4rZsjz18LDXOMHMB8GA1UdIwQYMBAAFIcF
EqIlNKiS9sy/WmplQQIG5SIJMA0GCSqGSIb3DQEBBQUAA4GBAGVu9/Gv2941Efvo
cCUB6RRE/yjaybLVmygS2fifnOTpCE4GEWcaMQsc63KEa1bbsoarKAPRqX5URMW2
xpcoy/6xIZN16cYf7+nwmN4Pc7TBdIer9ZxdfHhzE+PBkPWz7SH/JO5btGOUkw2/
UgZv7YvwSNgxtc891oYXYD4LB1YK
-----END CERTIFICATE-----
```

Table 11-5 Certificate file contents signed by CA (CERT.MQAIX2.arm)

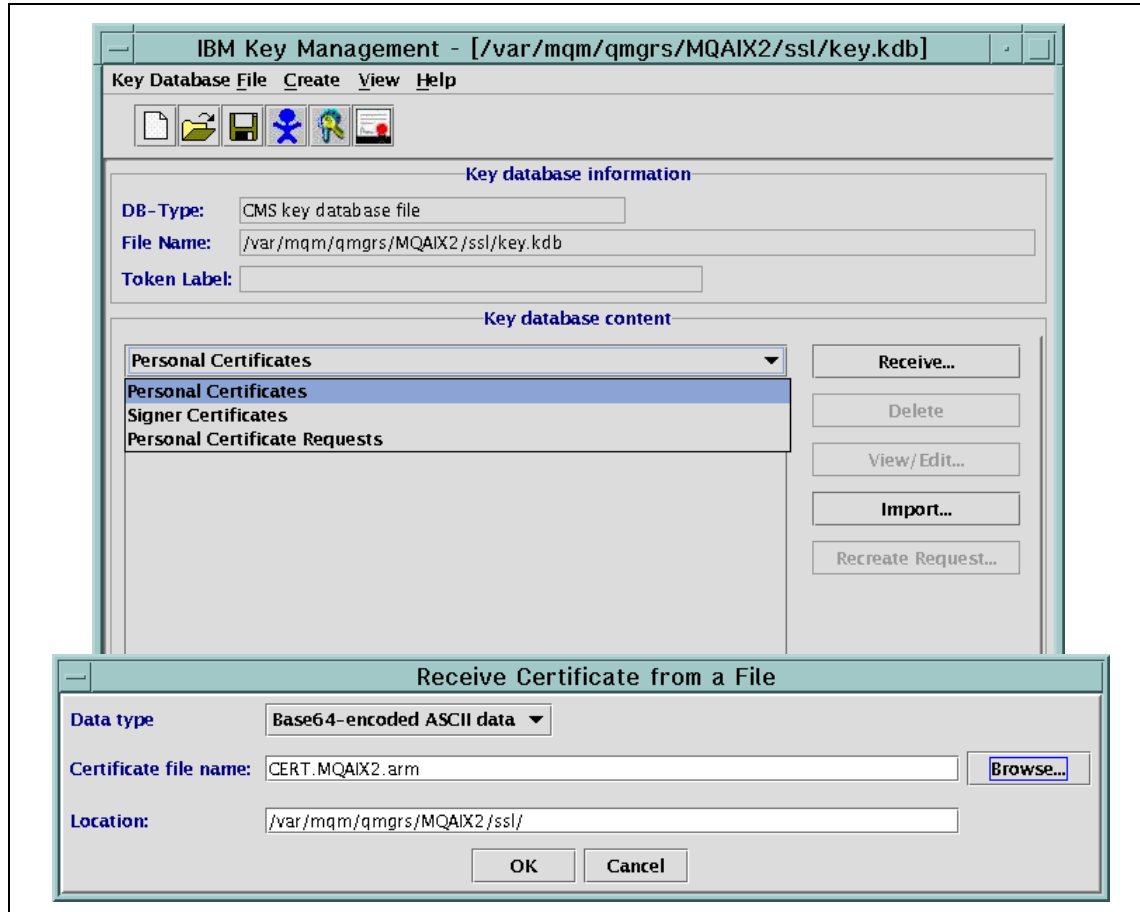


Figure 11-12 Receive the personal certificate signed by CA

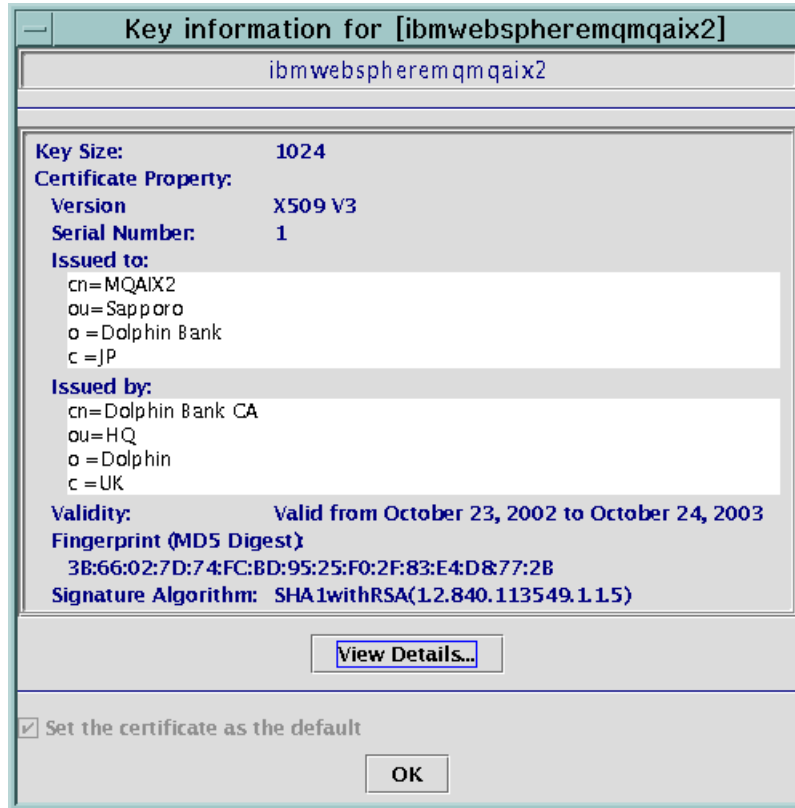


Figure 11-13 Key information of the imported private certificate

Exporting the CA signed certificate to the other machines

You can create a certificate request for all your machines on one particular machine by following the steps below.

1. Receive the CA signed certificate on the machine where the certificate request was created.
Select **Personal Certificates** and click **Receive....**
2. Export the key file: select **Personal Certificates** and click **Export/Import...**, then select the **PKCS12** key file type.

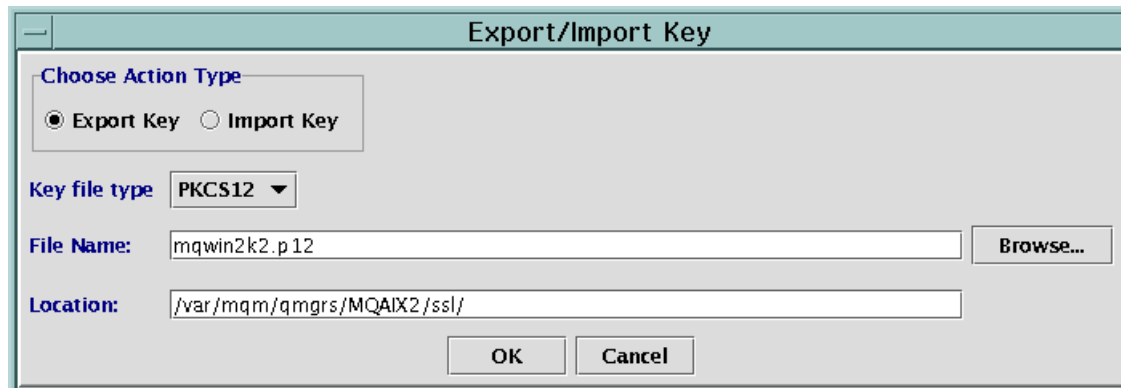


Figure 11-14 Export private certificate with the key

3. Copy these keys files to the other machines, for example, by FTP.

11.3.3 Preparing a certificate on Windows

This section explains how to create a key repository and prepare a certificate on Windows 2000.

Creating a key repository on Windows

WebSphere MQ provides MQ Explorer/Services and the `amqmcert` command to manage the key repository. The `amqmcert` command is useful for the Client environment because it does not have the WebSphere MQ GUI-based interface.

When a certificate is imported, the key repository, known as the *key store* in Windows, is automatically appended to the given location. In all cases, a suffix `.STO` is added.

Obtaining a CA signed certificate

Follow the steps described in “Creating a key repository on AIX” on page 206 when you use iKeyman.

Storing your certificate on Windows

You can import your certificate to the WebSphere MQ key store using WebSphere MQ Explorer or WebSphere MQ Services. You can choose to first import it to the Windows system store using the Microsoft certificate management; then you can use the WebSphere MQ certificate management to import it from the Windows system store to the WebSphere MQ key store. Alternatively, you can import it directly to the WebSphere MQ key store.

Using Microsoft Internet Explorer

1. Import your private certificate to the Windows system store by clicking **Tools** -> **Internet Options**. Choose the **Content** tab and select **Certificate**.

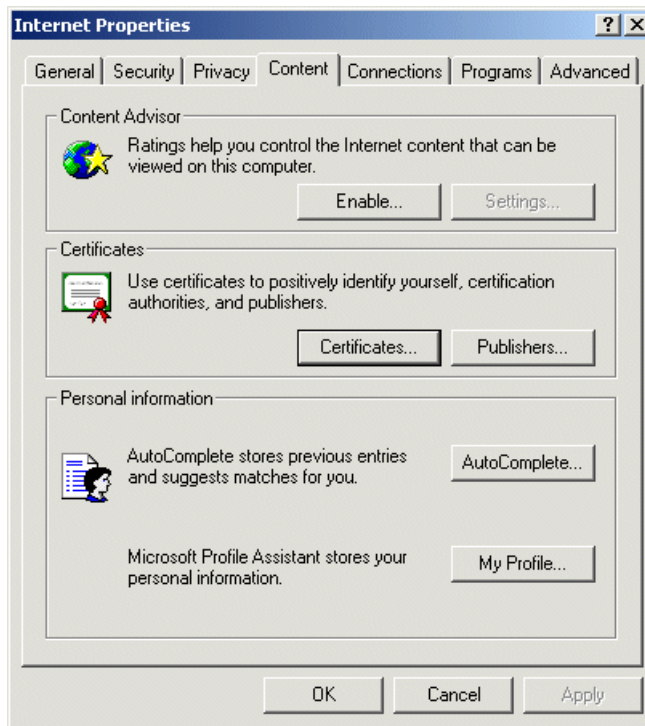


Figure 11-15 Internet properties

2. Import the CA signed certificate to the Personal tab.
3. Click **Import** and input the file name.
4. Input the password and select **Mark the private key as exportable**.

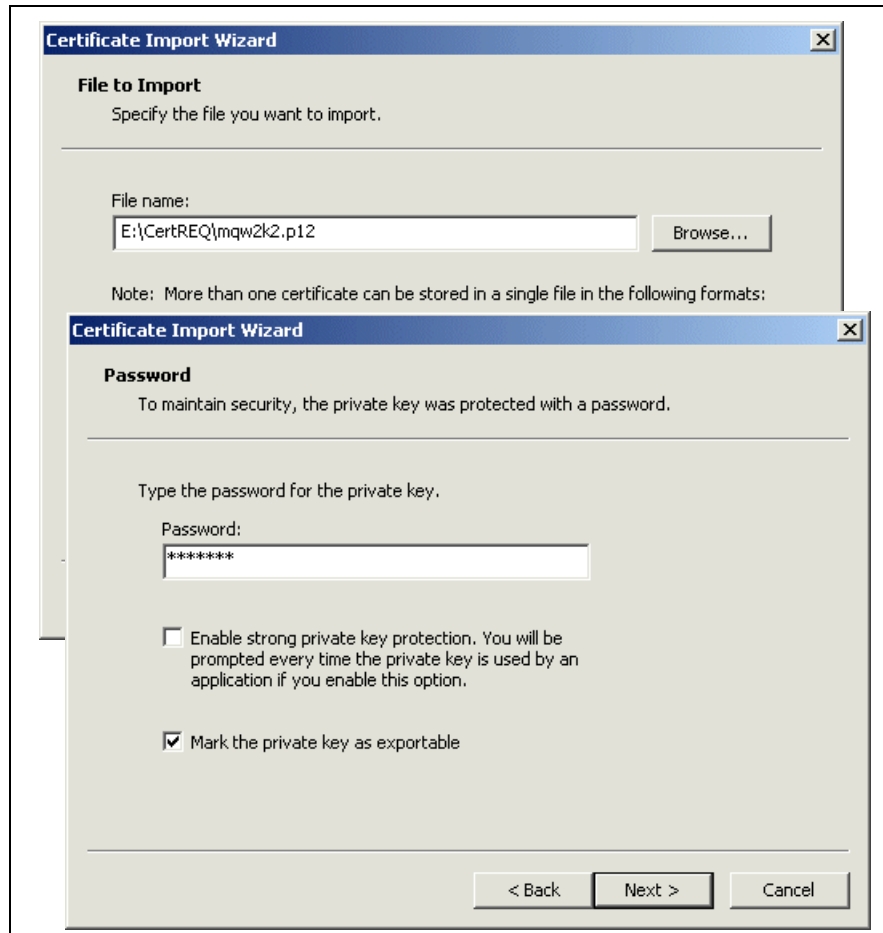


Figure 11-16 Import private certificate to Windows system store

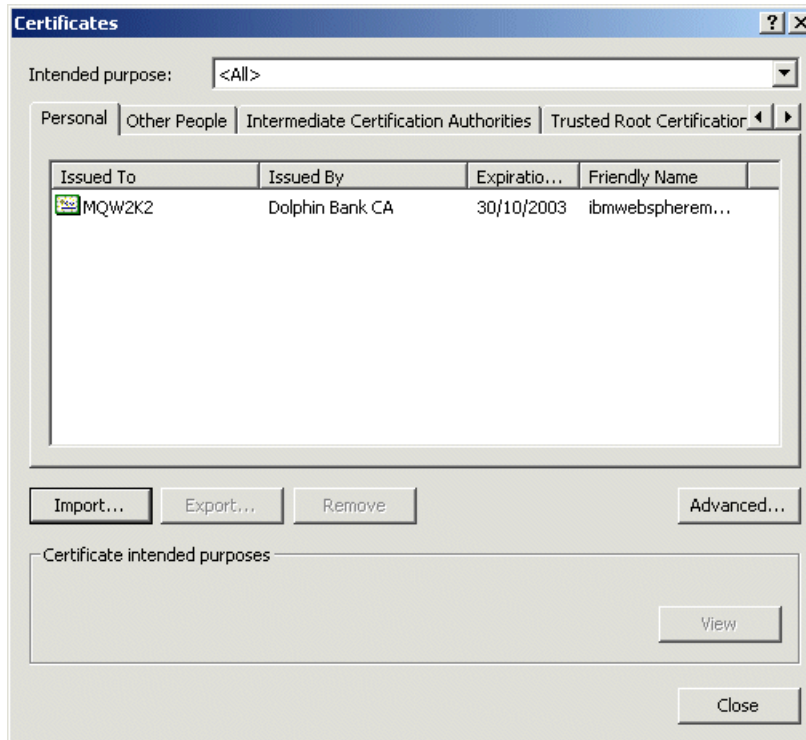


Figure 11-17 Imported certificate

Using WebSphere MQ Explorer

1. Add the CA's certificate by opening WebSphere MQ Explorer and expanding the Queue Manager folder.
 Select **Properties -> SSL -> Manage SSL Certificates -> Import from a file.**
 Enter your certificate file name and password

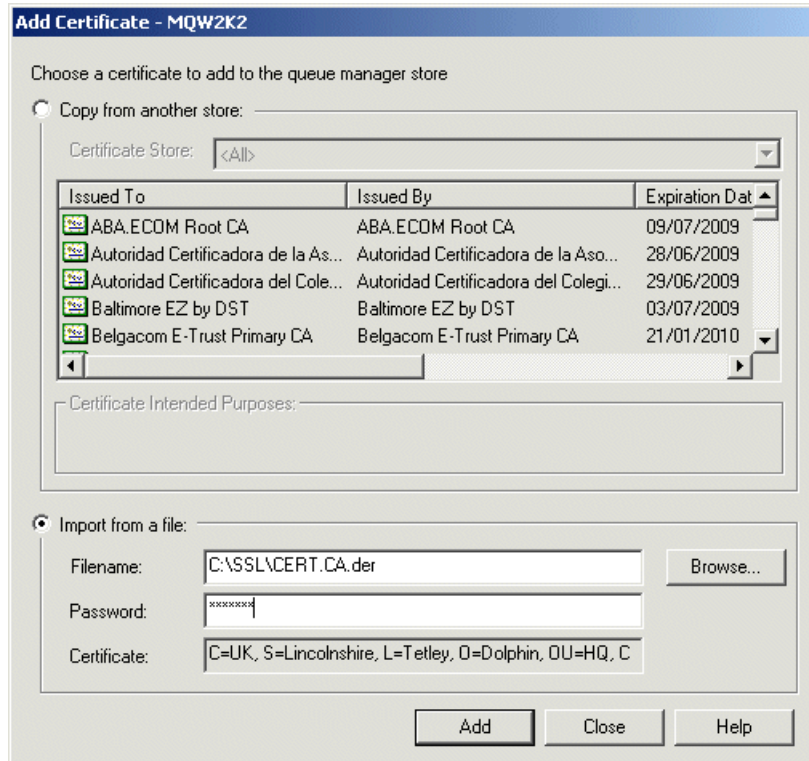


Figure 11-18 Import CA's certificate

2. Assign your certificate to WebSphere MQ queue manager by clicking **Assign** -> **Add**. Select your certificate.

Tip: Assigning a certificate is a special procedure for WebSphere MQ on Windows. It is not required for WebSphere MQ on AIX.

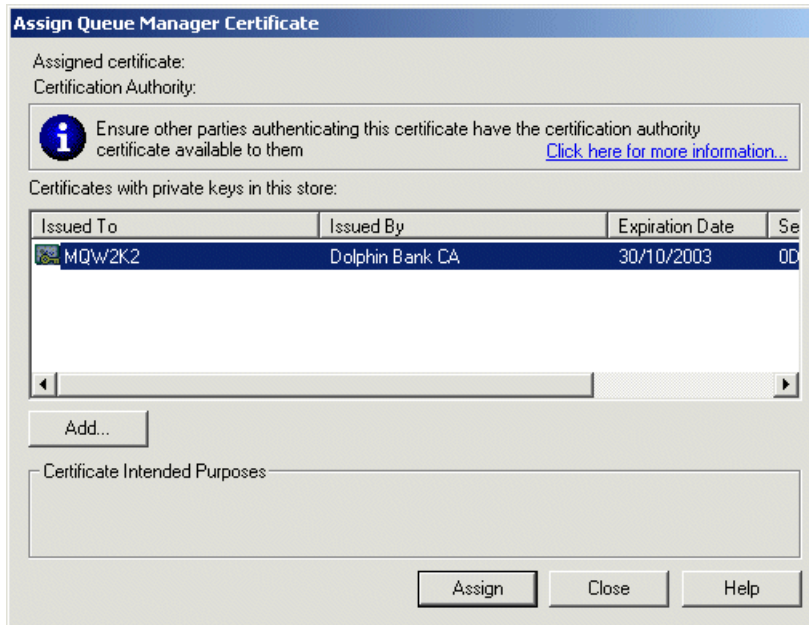


Figure 11-19 Assign certificate to a queue manager

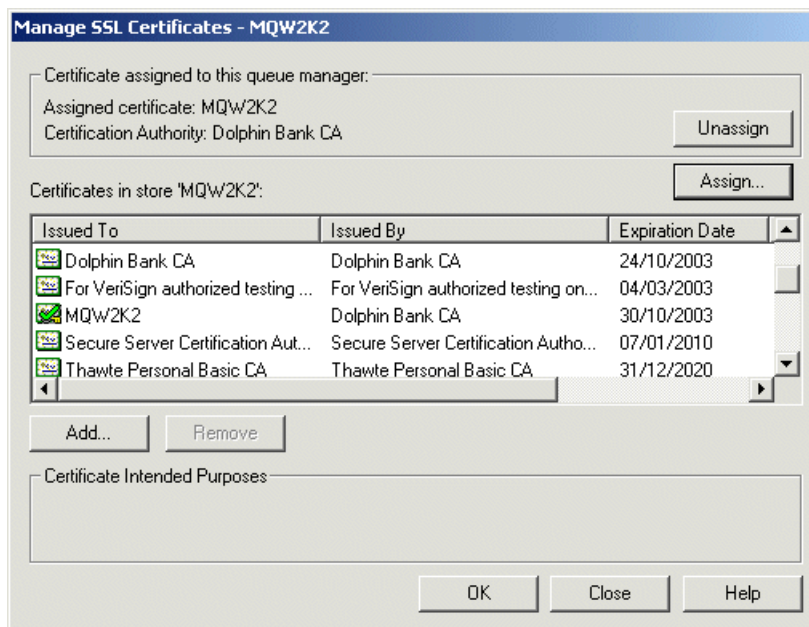


Figure 11-20 Assigned CA and private certificates

Tip: A check mark is added to an assigned certificate.

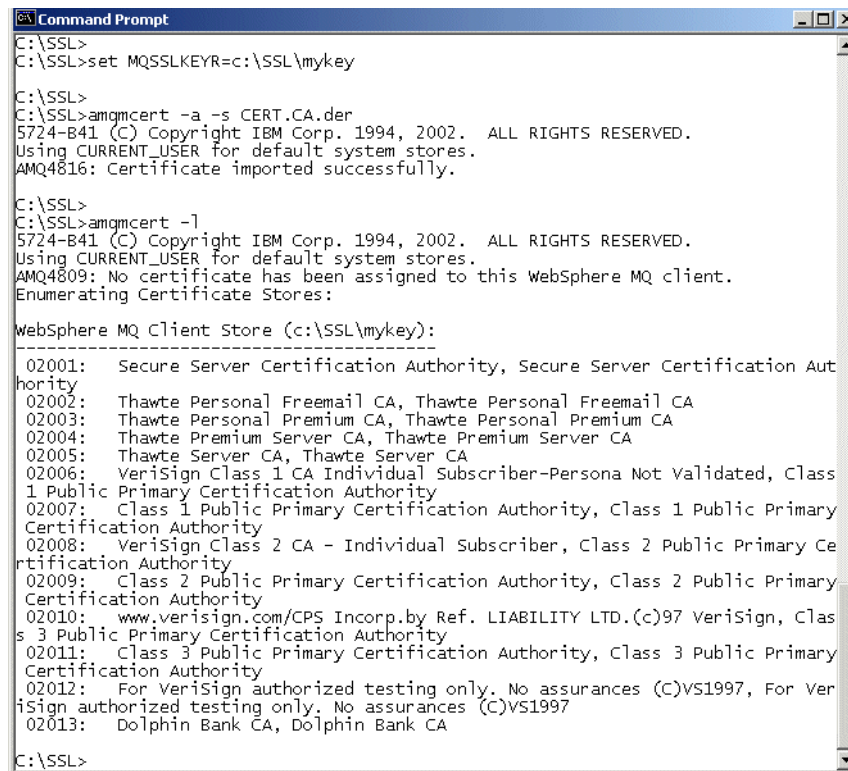
Storing your certificate on Windows with amqmcert (for client)

1. Set the MQSSLKEYR environment variable to specify your Key store name.

```
set MQSSLKEYR=c:\ssl\mykey
```

A file called mykey.sto is created.

2. Add the CA's certificate using the **amqmcert** command.



```
Command Prompt
C:\SSL>
C:\SSL>set MQSSLKEYR=c:\ssl\mykey
C:\SSL>
C:\SSL>amqmcert -a -s CERT.CA.der
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Using CURRENT_USER for default system stores.
AMQ4816: Certificate imported successfully.
C:\SSL>
C:\SSL>amqmcert -l
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Using CURRENT_USER for default system stores.
AMQ4809: No certificate has been assigned to this WebSphere MQ client.
Enumerating Certificate Stores:

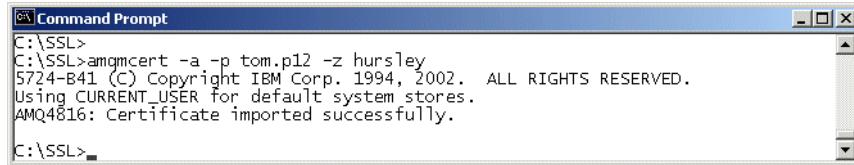
WebSphere MQ Client Store (c:\ssl\mykey):
-----
02001:   Secure Server Certification Authority, Secure Server Certification Authority
02002:   Thawte Personal Freemail CA, Thawte Personal Freemail CA
02003:   Thawte Personal Premium CA, Thawte Personal Premium CA
02004:   Thawte Premium Server CA, Thawte Premium Server CA
02005:   Thawte Server CA, Thawte Server CA
02006:   VeriSign Class 1 CA Individual Subscriber-Persona Not Validated, Class 1 Public Primary Certification Authority
02007:   Class 1 Public Primary Certification Authority, Class 1 Public Primary Certification Authority
02008:   VeriSign Class 2 CA - Individual Subscriber, Class 2 Public Primary Certification Authority
02009:   Class 2 Public Primary Certification Authority, Class 2 Public Primary Certification Authority
02010:   www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign, Class 3 Public Primary Certification Authority
02011:   Class 3 Public Primary Certification Authority, Class 3 Public Primary Certification Authority
02012:   For VeriSign authorized testing only. No assurances (C)VS1997, For VeriSign authorized testing only. No assurances (C)VS1997
02013:   Dolphin Bank CA, Dolphin Bank CA
C:\SSL>
```

Figure 11-21 Import CA's certificate

3. Add your personal certificate:

```
amqmcert -a -p tom.p12 -z hursley
```

tom.p12 is the personal certificate file name.



```
Command Prompt
C:\SSL>
C:\SSL>amqmcert -a -p tom.p12 -z hursley
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Using CURRENT_USER for default system stores.
AMQ4816: Certificate imported successfully.
C:\SSL>
```

Figure 11-22 Import personal certificate signed by CA

4. Assign your certificate to a WebSphere MQ client.

First, show a handle number of your certificate:

```
amqmcert -l
```

Then, assign it using the handle number:

```
amqmcert -d 02014
```

Tip: Assigning a certificate is a special procedure for WebSphere MQ for Windows. It is not required for WebSphere MQ for UNIX.

```

Select Command Prompt
C:\SSL>
C:\SSL>amqmcert -l
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Using CURRENT_USER for default system stores.
AMQ4809: No certificate has been assigned to this WebSphere MQ client.
Enumerating Certificate Stores:

WebSphere MQ Client Store (c:\SSL\mykey):
-----
02001: Dolphin Bank CA, Dolphin Bank CA
02002: For VeriSign authorized testing only. No assurances (C)VS1997, For VeriSign authorized testing only. No assurances (C)VS1997
02003: Class 3 Public Primary Certification Authority, Class 3 Public Primary Certification Authority
02004: www.verisign.com/CPS Incorp.by Ref. LIABILITY LTD.(c)97 VeriSign, Class 3 Public Primary Certification Authority
02005: Class 2 Public Primary Certification Authority, Class 2 Public Primary Certification Authority
02006: VeriSign Class 2 CA - Individual Subscriber, Class 2 Public Primary Certification Authority
02007: Class 1 Public Primary Certification Authority, Class 1 Public Primary Certification Authority
02008: VeriSign Class 1 CA Individual Subscriber-Persona Not Validated, Class 1 Public Primary Certification Authority
02009: Thawte Server CA, Thawte Server CA
02010: Thawte Premium Server CA, Thawte Premium Server CA
02011: Thawte Personal Premium CA, Thawte Personal Premium CA
02012: Thawte Personal Freemail CA, Thawte Personal Freemail CA
02013: Secure Server Certification Authority, Secure Server Certification Authority
02014: * tom, Dolphin Bank CA

C:\SSL>amqmcert -d 02014
5724-B41 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Using CURRENT_USER for default system stores.
Enumerating Certificate Stores:
Assigned MQClient Certificate:
    Name: tom
    CA: Dolphin Bank CA

C:\SSL>

```

Figure 11-23 Assign your private certificate

An asterisk (*) shows that this certificate contains a private key.

11.3.4 Preparing a certificate on z/OS

This section explains how to create the CA certificate on z/OS.

Creating the CA certificate on z/OS using RACF

To create the CA certificate on RACF that will be used to sign incoming certificate requests (top of the certificate chain), we used the following **RACDCERT** command. The **WITHLABEL** contents are used on the **GENCERT** command that you use when signing other certificates requests:

```

RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('Dolphin Bank CA') OU('HQ')
O('Dolphin') L('Tetley') SP('Lincolnshire') C('UK')) WITHLABEL('Dolphin
Bank CA') ICSF

```

ICSF tells RACF to use crypto hardware if it is set up and available. You can check that this has been done by listing the certificate using the following command:

```
RACDCERT CERTAUTH LIST(LABEL('Dolphin Bank CA'))
```

Signing the certificate request

In order to sign a certificate request with the CA certificate, you have to import the certificate request to a dataset on z/OS. For the business scenario, we used FTP to send the request across to the host on z/OS.

MQAIX2

The incoming certificate from MQAIX2 looks like the one in Figure 11-24.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrDCCARUCAQAwbDELMAkGA1UEBhMCSlAxETAPBgNVBAGTCEnhva2thaWRvMRAwDgYD
YVQKHEwdT.YXBwb3JvMRUwEwYDQKKEwxEb2xwaGluIEJhbmsxEDA0BgNVBAsTB1NhcHBvc
m8xDzANBgNVBAMT.Bk1RQUlyMjCBnzANBgkqhkiG9w0BAQEFAA0BjQAwgYkCgYEAj8qgBZ1KbHFde
jNp+hev1QdBx1ia.JELCU5xucGMWfzFPT32QC1MPCiJEce0tw+jwg8aRQXaH+VTad2qdMyH8aughBB
LPP0eURw982TKF.UAltsUquTd/aMK9Cm/zibEoFciEM0WuCmaZsTSRXZL+Qa1fm/pm471B5C8Le90CsJdUCAwEAAaAA
MA0GCSqGSIb3DQEBBAAUAA4GBAHKkZml0T8xfgjph5Hv0HWfaeKF/cgyupLr0FtUUQpgZwfwVQDmt
tAVYyn0MxBcFvc18tIi/Dgn/GA3QxahcG3s9JQtytcCu30dWokaqgAD4KjWLl0iSxr4t5eBCG1jZc
RWEtzMbnzafyWe0R9mcFDiGkrmtIE27XLZAhfU7MUJT
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-24 Incoming certificate from MQAIX2

However, when the certificate is browsed as shown in Figure 11-24, you can see that there is a problem: there is an additional character at the end of several lines. If looked at using HEX, you can see that it is an “0D” or carriage return character. If we leave this here, it will cause the signing of the certificate to fail. So we edit the dataset while showing it in HEX as containing the imported certificate and remove the spurious “0Dx” characters by replacing them with “40x”. Once edited, the certificate request looks like the one shown in Figure 11-25

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrDCCARUCAQAwbDELMAkGA1UEBhMCSlAxETAPBgNVBAGTCEnhva2thaWRvMRAwDgYD
YVQKHEwdT.YXBwb3JvMRUwEwYDQKKEwxEb2xwaGluIEJhbmsxEDA0BgNVBAsTB1NhcHBvc
m8xDzANBgNVBAMT.Bk1RQUlyMjCBnzANBgkqhkiG9w0BAQEFAA0BjQAwgYkCgYEAj8qgBZ1KbHFde
jNp+hev1QdBx1ia.JELCU5xucGMWfzFPT32QC1MPCiJEce0tw+jwg8aRQXaH+VTad2qdMyH8aughBB
LPP0eURw982TKF.UAltsUquTd/aMK9Cm/zibEoFciEM0WuCmaZsTSRXZL+Qa1fm/pm471B5C8Le90CsJdUCAwEAAaAA
MA0GCSqGSIb3DQEBBAAUAA4GBAHKkZml0T8xfgjph5Hv0HWfaeKF/cgyupLr0FtUUQpgZwfwVQDmt
tAVYyn0MxBcFvc18tIi/Dgn/GA3QxahcG3s9JQtytcCu30dWokaqgAD4KjWLl0iSxr4t5eBCG1jZc
RWEtzMbnzafyWe0R9mcFDiGkrmtIE27XLZAhfU7MUJT
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-25 Certificate request from MQAIX2 after editing

The imported certificate is now ready to be signed.

Signing an incoming request using a given dataset creates a CA signed certificate. LABEL on signwith is the label from the CA certificate. WITHLABEL is the label this certificate is known as.

Once signed by the CA using the following command:

```
RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.MQAIX2.ARM') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA')) WITHLABEL('DB MQAIX2 CA Signed')
```

the certificate looks like that shown in Figure 11-26.

```
-----BEGIN CERTIFICATE-----
MIIC4zCCAkygAwIBADANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJVSzEy
MBMGA1UECBMmTGlUy29sbnNoaXJlMQ8wDQYDVQQHEwZUZXRzZXkxEDA0BGNVBAoT
B0RvbHBBoaw4xCzAJBgNVBAsTAKhRMRRgwFgYDVQQDEw9Eb2xwaGlueJhbmsgQ0Ew
HhcNMDIxMDIzMDQwMDAwWWhcNMDMxMDIOMDM1OTU5WjBuMQswCQYDVQQGEwJVSzEy
MBMGA1UECBMmTGlUy29sbnNoaXJlMQ8wDQYDVQQHEwZUZXRzZXkxEDA0BGNVBAoT
B0RvbHBBoaw4xCzAJBgNVBAsTAKhRMRRgwFgYDVQQDEw9Eb2xwaGlueJhbmsgQ0Ew
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAK7cxFPd9V3PdPnLWZggA114jEKY
KIZ572chCgdIYakY6R9BFaRfmpvbecToGpuS3BETW54GcviZEJoGtswKZfeFh4d7
Mjx+kDgPnV230uze25VmJ0ZP2JTFBLAXy8FGmGU6M4D/z9aqwp0r1SwH8k6IiIh
a4+ZxGkRbUx1d9nAgMBAAGjgZAwgY0wSwYJVR0PAYb4QgENBD4TPEdlbmVuyXRl
ZCBieSB0aGUU2VjdxJlV2F5IFNlY3VyaXR5IFNlcnZlcjBmb3Igt1MvMzkwIChS
QUNGKTA0BgNVHQ8BAf8EBAMCAAYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFggQU
giUSoiU0qJL2zL9aamVBAbgllgkwdQYJKoZIhvcNAQEFBQADgYEAHC9VxPGVK8yY
/87w96hu9bNn0LLUZPmZBhFsyxGGccCMvkibdY26Ven54F4Hutug9NgsVhIgt59f
ReGvE79LWgE6ssl0Y9oNq27+vtGCLpuF5m2mzh2yKjXKUK+j0oSRYCitrh4ecUu
f7H0wcaMcf9RvKjckaiBj9QrBHdImqt=
-----END CERTIFICATE-----
```

Figure 11-26 CA signed certificate for MQAIX2

MQAIX3

We had a similar problem with the imported certificate from MQAIX3, so we edited this file in the same way.

The incoming certificate from MQAIX3 looks like Figure 11-27.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCR0IxFtATBgNVBAgTDExpbnNvbG5zaGlueJZTEPMA0GA1UE
BxMGVGV0bG95SMRUwEwYDVQQKEwEeb2xwaGlueJhbmsgDzANBgNVBAsTBURldGxleTEPMA0GA1UE
AxMGTlVFBsVgZMIgFMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQClfJ6tjnwHhtZK3W7NPQp5BQx.
aC9W9fx/VZ4DfMxjJAcVBGPJns6cIbMT3PoqzZgfQYu4xlQnTiiYmpvMYTjafAZi7cuJhefar64h.
06HMwaBHfJmEuuGYJwm0ngmFdxTxP1NmUISmh1VT7gxwxEpZXukqSk5D72MdKJ28CUucQIDAQAB.
oAAwDQYJKoZIhvcNAQEEBQADgYEABNU2jbw7ABM5Zx84IjJEi0GiUB/SME2W2xY8nHzBgWpKteJZ.
JTNmziRtYZAxCFHqQH0CxmrmhoxW8hgUCzhst0idNYmTL9gRUX40yn/Hk0YM4j6NsvFEIxsne2U7.
rRyAn5dMSGBaj0EFX9+wnTmyUCHAarm7yf6dJZKfw9rJ7fE=.
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-27 Incoming certificate from MQAIX3

Once edited, the certificate looks like the one in Figure 11-28.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCR0IxFtATBgNVBAgTDExpbnNvbG5zaGlueJZTEPMA0GA1UE
BxMGVGV0bG95SMRUwEwYDVQQKEwEeb2xwaGlueJhbmsgDzANBgNVBAsTBURldGxleTEPMA0GA1UE
AxMGTlVFBsVgZMIgFMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQClfJ6tjnwHhtZK3W7NPQp5BQx.
aC9W9fx/VZ4DfMxjJAcVBGPJns6cIbMT3PoqzZgfQYu4xlQnTiiYmpvMYTjafAZi7cuJhefar64h.
06HMwaBHfJmEuuGYJwm0ngmFdxTxP1NmUISmh1VT7gxwxEpZXukqSk5D72MdKJ28CUucQIDAQAB.
oAAwDQYJKoZIhvcNAQEEBQADgYEABNU2jbw7ABM5Zx84IjJEi0GiUB/SME2W2xY8nHzBgWpKteJZ.
JTNmziRtYZAxCFHqQH0CxmrmhoxW8hgUCzhst0idNYmTL9gRUX40yn/Hk0YM4j6NsvFEIxsne2U7.
rRyAn5dMSGBaj0EFX9+wnTmyUCHAarm7yf6dJZKfw9rJ7fE=.
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-28 Certificate request from MQAIX3 after editing

The certificate is now ready to be signed. Once signed by the CA certificate using the following command:

```
RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.MQAI3.ARM') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA')) WITHLABEL('DB MQAI3 CA Signed')
```

the certificate looks like the one in Figure 11-29.

```
-----BEGIN CERTIFICATE-----
MIIC4zCCAkygAwIBAgIBBjANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJVSzEy
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDYGQHEwZUZXRsZXkxEDA0BgNVBAoT
B0RvbHBoaW4xZCZlY29sbnNoaXJlMQ8wDQYDYGQHEwZUZXRsZXkxZDAtBjEw
HhcNMDEwMDIzMDQwMDAwWWhcNMDMxMDI0MDM1OTU5WjBuMQswCQYDVQQGEwJH
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDYGQHEwZUZXRsZXkxZDAtBjEw
DERVbHBoaW4xZCZlY29sbnNoaXJlMQ8wDQYDYGQHEwZUZXRsZXkxZDAtBjEw
gZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKV8nq20fAeG21krdbS09CnkFDfO
L1b1/H9VngN8zGmKbXUEY8mezpwhsXpC+irNmB9Bi7jGVCdPwJiam8xh0Np8BmL
tY4mF59qvrIE7oczBoEd8kmYS64ZgnCbSeCYV3FPE/U2ZQ1yaHVVPuDHDESle6Sp
KTKPvYx0onbwJSSxAgMBAAGjZAwGYP0wSwYJYIZIAYb4QgENBD4TPEdlbmVYXRRL
ZCBieSB0aGUgU2VjZDZlY29sbnNoaXJlMQ8wDQYDYGQHEwZUZXRsZXkxZDAtBjEw
QUNGKTAdbGgNVHQ4EFggQUfuGaR3MPNlBPaiAXiBtsI7yWlyAwHwYDVR0jBBgwFoAU
gIUsoIU0qJL2zL9aamVBAGblIlgkwdQYJKoZIhvcNAQEFBQADgYEAKEGFUuYJKPuB
6PH035+HfEeCtK4Lsztxws+csN7GjF6EwKQ03WCjrnqwf13Jq8gh831gmw3MQ0ab
ePD600hP1oQ0m+PfgL+/J4Ei+tiCkMAYQIr84YW09PQcgUKA2qzbYvT3zpn0c5k
U9AEedXTxe7CdeEtR+QmMptcrP5DAfwJ=
-----END CERTIFICATE-----
```

Figure 11-29 CA signed certificate for MQAI3

MQ400

The incoming certificate from MQ400 looks like the one in Figure 11-30.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBvDCCASUCAQAwfDELMaKGA1UEBhMCR0IxEtAPBgNVBAgTCFNIRVRMQ5EMREw
DuYDVGQHEwTSEVUTEFORDELMBQGA1UEChMNU0hFVExBTKQgQkFOSzEfmB0GA1UE
CxMlU0hFVExBTKQgQkFOSyBTRUNUUKLUWTEOMAwGA1UEAxMFTVE0MDAwgZ8wDQYJ
KoZIhvcNAQEBBQADgY0AMIGJAoGBAKGzPmWmYSXon+vLInU0hRyZsJ3pFo6na8P6
W40Kz7zNq+Lbfl+nXy4G7z+D3/GQpI6BGocBH21QuJoL2XWt5/XmZvPHGEZ5Jkf
tuzjVytRBg+DRqbT9LECTJV0LOR9Mg7Y+jPe1A3XVJgfAxRMBvH8ljz/xvJailRT
s3hjd1J/AgMBAAGgADANBgkqhkiG9w0BAQUFAA0BgQA4+ySQaAF2D3+mKYbjZsT
bDZMGiJQwbIvvlbhtxzBU08UkmmsHJxigRrC/0BpLRTLg/MSlfMvyNfvTItYgUW
k/ph41fhiJjX4L0dixryD/Z0MJbeNVzGvLVaQSTVZ6pcGbaZPeLWrpERQqxj0szA
PzbLqLVjNs3UvJ4zf/1xPw==
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-30 Incoming certificate request from MQ400

When trying to sign with the CA certificate using the following command:

```
RACDCERT ID(SDRES) GENCERT('SDRES4.MQ400.CERTREQ') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA')) WITHLABEL('DB MQ400 CA Signed')
```

this error message came up:

The certificate that you are creating has an incorrect date range. The certificate is added with NOTRUST status.

so we changed the state to TRUST using the following line:

```
RACDCERT ID(SDRES) ALTER (LABEL('DB MQ400 CA Signed')) TRUST
```

and tried signing it again. This time it worked.

Once signed, the certificate looked like the one in Figure 11-31.

```
-----BEGIN CERTIFICATE-----
MIIC8TCCAlqgAwIBAgIBCTANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJVSzEy
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDVQQHEwZUZXRsZXkxEDA0BgNVBAoT
B0RvbHBoaW4xZCZAJBgNVBAsTAkhrMRgwFgYDVQQDEw9Ebz2xwagluIEJhbmsgQ0Ew
HhcNMDIxMDI0MDQwMDAwHhcNMDMxMDI1MDM1OTU5WjB8MQswCQYDVQQGEwJHqJER
MA8GA1UECBMIU0hfVExBTkQxETAPBgNVBACtCFNIRYRMQU5EMRYwFAYDVQQKEw1T
SEVUTEFORCBCQUSLMR8wHQYDVQQLExZTSEVUTEFORCBCQUSLIFNFQ1VSSVZRMQ4w
DAYDVQQDEwYNU0hVZmDCBnzANBgkqhkiG9w0BAQEFAA0BjQAwYkCgYEAob0kzAxh
Jeif68sidTSFhJmwnekWjqdrw/pbjQrPvM2r4tt8v6dfLgbvP4Pf8ZCKjoeIahwe
fbVC4mgvZda3n9eZm880YRnkMR+270NXXK1EGD4NGptP2UQJMLU6U5H0ydtj6M97U
DddUmB8DFEwG8fyWPP/G8lqKYFOzeGN3Un8CAwEAAa0BkDCBjTBLBgLghkgBhvC
AQ0EPHMR2VuZXJhdGVkIGJ5IHRoZSBTZW51cmVYXkGU2VjdXJpdHkgU2VydMvYy
IGZvc iBPuY8z0TAgKFJBQ0YpMBOGA1UdDgQWBBRpZEcFgi4hIALpiXvMDpw/0/Oi
fTfBgnVHSMEGDAwGBSAHRK1JTSokvbMv1pqZUECBuUICtANBgkqhkiG9w0BAQUF
AA0BgQcmmowcXtCRMLO/q2cRYTj fFEa0xz l91u6ph1jymS3v fBYvQ0zJbmU9i74D
NZAuw6FWIi17Dr gzN/hv/EXwo2NAv4AJvfH9k3P7HC3YF6Jqe149M5BsvgMuYjc3
U562K16DiC9B2Bz0ohr1hzrPxrR1Wuj+e+Ukw2snp+RjYyWqLE==
-----END CERTIFICATE-----
```

Figure 11-31 CA signed certificate for MQ400

MQW2K2 (Windows server)

The incoming certificate from MQW2K2 looks like the one in Figure 11-32.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqTCCARICAQAwjELMAkGA1UEBhMCQVUxETAPBgNVBAGTCFZpY3RvcmlhMQ8wDQYDVQQHEwZG
b3N0ZXIxFtATBgNVBAoTDERvbHBoaW4gQmFuazEPMA0GA1UECzMGRm9zdGVyMQ8wDQYDVQQDEwZn
UVVycyZlwgZ4wDQYJKoZIhvcNAQEBBQADgYwAMIGIAoGAbxLXjlnfykVc9Wg3oN7F99AqHqTWA sSN.
nTAoXpop9h5dPjITV+1eq2/GxcezqdvMJ1dJelNo0vpVT1QsARATrt0HL4qfh+2UsKbjfNKfyLUK5.
CSqGS Ib3DQEBAUAA4GBABsGaJ1dZh9WJoIO0MTZ/52LCvZq9PrLchPZCjRoc8S3BJfvxmQMBsbC.
EOhGsESx5xql54pl7yG5/sCKIg9E3gl+hnFulRePCJn6AEDyFezC4BrdYfsFMzmf+8tBNI0l0JgI.
Lmyv+dPFEUPoiGay+c70x09/wp5CTIehX2F4B9bU.
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-32 Incoming certificate request from MQW2K2 before edit

This too has additional characters at the end of the lines and needs to be edited.

Once edited, the certificate looks like the one in Figure 11-33 on page 229.


```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBqTCCARICAQAwwJELMAkGA1UEBhMCQVUxEAPBgNVBAGTCFZpY3RvcmlhMQ8wDQYDVQQHEWZG
b3R0ZXIxFTATBgNVBAAoTDERvbHB0aW4gQmFuazEPMA0GA1UECzMGRm9zdGVyMQ8wDQYDVQQDEWZn
UUVcySzUwZ4wDQYJKoZIhvcNAQEBBQADgYwAMIGIAoGBAbXlXjWnfYkVc9Wg3oN7F99AqHqTWAaSN
nTAoXpop9h5dPjI7V+1eq2/GxcezqdvMJ1dWeNo0vpVT1QsARARt0HL4qfH+2UsKbjfNKfyLUK5
vaqaS5U5QmZA/od5R5i3ahhsYbJDI3Hh3cUL4gArdq6Cj7/i+9PwxIMvoAizdKMCaWEAAaAAMA0G
CSqGSIb3DQEBBAAUAA4GBABsGaJ1dZh9WwIo0MTZ/52LCvZq9PRLchPZCjRoc8S3BJfvxm0MBsbC
EOhGsESx5xql54pl7yG5/sCKI9e3gL+hnFulRePCJn6AEDyFezC4BrdYfsFMzmf+8tBNI0loJgI
Lmyv+dPFEUPoiGay+c70x09/wp5CTIehX2F4B9bU
-----END NEW CERTIFICATE REQUEST-----

```

Figure 11-33 Certificate request from MQW2K after editing

Once signed by the CA certificate using the following command:

```

RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.MQW2K2.ARM') SIGNWITH(CERTAUTH
LABEL('Dolphin Bank CA')) WITHLABEL('DB MQW2K2 Server CA Signed')

```

the certificate looks like the one in Figure 11-34.

```

-----BEGIN CERTIFICATE-----
MIIC3jCCAKEgAwIBAgIBDTANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEWJVSzEy
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDVQQHEWZUZXRzZXkxEDAOBgNVBAoT
B0RvbHB0aW4xZmF1b3R0ZXIxFTATBgNVBAsTAkhrMRgWFGYDYQDDEW9Ebz2xwaGluIEJhbmg0Ew
HhcNMDIxMDI5MDUwMDAwWhcNMDMxMDMwMDQ1OTU5WjBqMqswCQYDVQQGEWJBYTER
MABGA1UECBMIWmljdg9yaWExDzANBgNVBACTBkZvc3RlcjEYMBMGA1UEChMVRGR9S
cGhpb1BCYw5rMQ8wDQYDVQQLEWZGbzN0ZXIxZDZANBgNVBAMTBk1RVzJlMjBjCjBn
BkqhkiG9w0BAQEFAA0BjAAwgYgCgYBvGVeNad/KRVz1aDeg3sX30CoepNYCxi2D
MChemIn2Hl0+Mjtx7Y6rb8bFX70p28wnV1Z42g6+lVPYcWBEB0u3Qcvip+H7ZSwp
uN80p/IQrm9qppLlLCZkd+gPLHmLdqGGxhskMjcefdxQviACT2roKPV+L70/DG
Iy+gCLN0owIDAQABo4GQMIGNMEsGCWCGSAGG+EIBDQ+ExZHZW5lcmF0ZwQgYnkG
dGhlfNLY3VyZVgheSBTZWN1cmloesBTZXJ2ZXIgzZm9yIE9TLzMSMCAoUKFDRikw
HQYDVR00BBYEFNGceGLd+K/b4Xgg0N6AM/ascYBaMB8GA1UdIwQYMBaAFICFEqIL
NKiS9sy/WmplQQIG5SIJMA0GCSqGSIb3DQEBBQUAA4GBA1pv3fgK5BnZCg64NinB
MZiY64jK47JYhrizD6wtfCfPFR/t5gT+14px2kX5vEaI8LhSAT85Umc1VMVxDEKD
vxrsrC0t7drLbG5jZk85pUq6tB7dkh89Qo4w15Evm3dwo1NLSQMUQN1Sws/5ybiP
/BJLnY2h+YjKwrX/zjd9cm7B
-----END CERTIFICATE-----

```

Figure 11-34 CA Signed certificate for MQW2K2

MQW2K5 (Windows server)

The incoming certificate from MQW2K5 looks like the one in Figure 11-35.

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB1zCCAUAQAQAwwZyXCzAJBgNVBAYTAVTMRiWEAYDVQQREwlaSVAXMjM0NTYx
EDAOBgNVBAGTB0luZGllbmcEzEDAOBgNVBACTB0luZGllbmcEzIzAhBgNVBAoTGkRv
bHB0aW4gQmFuayBjbnRlcm5hdGlvbmFsmRkWFwYDYQDDEW9Ebz2xwaGluIEJhbmg0Ew
S5U5jMQ8wDQYDVQQDEWZnUUVcySzUwZ4wDQYJKoZIhvcNAQEBBQADgY8AMIGIAoGB
ALD0lo1FJJC4Yl44EZCbnSy2laRINziV6vtIWM0izkM45VKVXyq7ikfNJBt63/AuF
/mqlkhcqjspi9c1kDeHuZgU5DboKpBnYyP9w6MfN4JpvU3FHbnwfx14dAFi86cw2
2/g8NfK4xVGJN08d7s+sMKRpDdkcfm8V0dx0eJpYipdbAgMBAAGgADANBgkqhkiG
9w0BAQUFAR0BgQA6SrPLZ2gYslwCw8YlurQwkR8+UrGbDxqJFwmuLN7YgZlwz7urg
rVyh0TYbV9hk/WuZfhfFyHIXJfLWU8NC0vraMtl0oY9w62IHtrXH1St1JVjdyd
VZcNYBBp0B+sgSub8NgVBx3KYh5ffTBy871L9X/Z8Rh5Tzn6IyLL0omMba==
-----END NEW CERTIFICATE REQUEST-----

```

Figure 11-35 Incoming certificate request from MQW2K5

Once signed by the CA certificate using the following command:

```
RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.MQW2K5.ARM') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA')) WITHLABEL('DB MQW2K5 Server CA Signed')
```

the certificate looks like the one in Figure 11-36.

```
-----BEGIN CERTIFICATE-----
MIIDDDCCAnWgAwIBAgIBDjANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEwJVSzEy
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDZDQHEwZUZRZSXXkxEDA0BgNVBAoT
B0RvbHBoaW4xZm9uY29sbnNoaXJlMQ8wDQYDZDQHEwZUZRZSXXkxEDA0BgNV
HhcNMDIxMDU1MDUwMDAwHhcNMDMxMDUwMDQ1OTU5WjCBLjELMAKGA1UEBHMCMV
EjAQBgNVBETCVPpJUEEgMzQ1NjEjEQMA4GA1UECBMH5kaW5kaW5kaW5kaW5kaW
SW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW5kaW
BgNVBAsTEERvbHBoaW4xZm9uY29sbnNoaXJlMQ8wDQYDZDQHEwZUZRZSXXkX
hkIG9w0BAQEFAA0BjQAwgYkCgYEA5ajUUKKLHixjgRkJs2zLaVpEg30JXq+0hYz
SLOqzj1UpYdiruKR80kFPrf8C4X+aqWSFyq0ymL1zWQN4e5mBTkNugqkGdjI/3Do
x83gmm9TcUdufB9fXh0AWLzpzDbb+Dw0WTjFUYk3Tx3uz6wppGkN2Rx+bxXR3HR4
mlikl1sCawEAAa0BKDCBjTBLBglnhgBhvhCAQ0EPHMBR2VuZXJhdGVkIGJ5IHRo
ZSBTZWN1cmVYXkkgU2VjdXJpdHkgU2VydmlvYjZvcjBPUy8zOTAgKFJBQ0YpMB0G
A1UddGgWBBtu/X3rCL+Q3hhjG91ytxE8LZ2p3TAFBgNVHSMEGDAwGBSAHRK1JTSo
kvbMv1pgZUECBWUjCTANBgkqhkiG9w0BAQUFAA0BgQBfsoWviT3847byTb9Xns1
/kUUX/FPQwrwrEsRD08PRQZjwBpn16JHDpc4DTA952KujISZ60cTgyZHD4ZEKKmiN
ZefZubP4jyimpDv0/QhyNA33eUQ5UQ0rjV2cmSoPmSgRbwU06fdeW0JGipUdsdy4n
0yzUbJp+uuSjEN0hLE0H4k==
-----END CERTIFICATE-----
```

Figure 11-36 CA signed certificate for MQW2K5

TOM (Windows client 1)

The incoming certificate from Windows client TOM looks like the one in Figure 11-37.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBcTCB2wIBADAgMQswCQYDVQQGEwJHJQjEYVMBMGA1UEChMRRG9scGhpbjBCYw5r
MQwwCgYDVQQDEwN0b20wZ28wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAKk1cZnz
FXvU3Xy9nyZxtl4t+XvBTFmE6p+FT/RLznMVU5Iic5mqBEPTCrYbWslLlrUUsv+0
Qd3A2SFsSl+sYwEQF2MNS67UrJ+ZYj0L8BFj0SK0iQsqbPh07HLCjnI06TRUnEt
PrfRtj5prQg/vG1CxJHd0z6/ROfhrYn/UdlzAgMBAAGGADANBgkqhkiG9w0BAQQA
AA0BgQAQ71BScYw0u/xU5eE9XIJT9+wGDqJaRhb72Ydcb7UhhIEBjY3ipjH4pb1n
sMsrI6Gzqnm60L7tc4Y1ZauNqwCv54tga5R9DCG5GL3GvNEwVxrMBpcnMrqsF1S
Kw9Zo2H1GoMM3zl9kZAK741BaCkrqwF84r4Dasm1Rfli/AjcaA==
-----END NEW CERTIFICATE REQUEST-----
```

Figure 11-37 Incoming certificate request for Windows Client 1 TOM

Once signed by the CA certificate using the following command:

```
RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.TOM.ARM') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA')) WITHLABEL('DB MQClient1 CA Signed')
```

the certificate looks like the one in Figure 11-38 on page 231.

```

-----BEGIN CERTIFICATE-----
MIICpzCAAhCgAwIBAgIBAZANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEWJYSzEY
MBMGA1UECBMmTGluY29sbnNoaXJlMQ8wDQYDVQQHEWZUZXRrsZXkxEDA0BgNVBAoT
B0RvbHBoaW4xZCZAJBgNVBAsTAkhrMRmgwFgYDVRQQDEw9Ebz2xwaGluIEJhbmsgQ0Ew
HhcNMDIxMDIzM0QwMDAwHhcNMDMxMDIOMDM1OTU5WjAgMQswCQYDVQQGEWJHJHJjEY
MBMGA1UEChMmRz9scGhpbjBCYw5rMQwwCgYDVRQQDEwN0b20wgZ8wDQYJKoZIhvcN
AQEBBQADgY0AMIGJAoGBAKk1cZnzFXvU3Xy9nyZxtl4t+xVbTFmE6p+FT/RLznMV
U5Iic5mqBEPTCrYbWslLlrUUsv+0Qd3A2SFsSl+sYwvEQF2MNS67UrJ+ZYj0L8BF
j0SKoiQsqbPh07HLCjnI06TRUnEtPrfRtj5SprQg/vG1CxJHd0z6/R0FhryN/UdWz
AgMBAAGjgZAwgY0wSwYJYIZIAyb4QgENBD4TPEdlbmVYXRlZCBieSB0aGUU2Yj
dXJlV2F5IFNlY3VyaXR5IFNlcnZlcjBmb3Igt1MvMzkwIChSQUNGKTAdBgNVHQ4E
FgQUxcks/oIugz3PmGG1gsqHRoYPRzswHwYDVR0jBBgwFoAUgiUSoIU0qJL2zL9a
amYBAgblIgwDQYJKoZIhvcNAQEFBQADgYEAADp+BQ5Tg/y+jh730fiflCMGpNze
U8sPf+5Jyf+R/GHt8tPMZ3fMxZMYxLZ6k2EUTx0y07Ynkz0PCw/2VoFchkr0GYxq
ikqibm+qmIS7ZOM1wByY/hbZ87gYwdQzK/sfGwki1WRATdLhH1BNyH8QR0JFngW
BWH/4mk9b/2SU55=
-----END CERTIFICATE-----

```

Figure 11-38 CA signed certificate for Windows Client 1 TOM

ALICE (Windows client 2)

The incoming certificate from Windows client ALICE looks like the one in Figure 11-39.

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB5zCCAVACAQAwwaYxCzAJBgNVBAYTALVTRMRwEQYDVQQREwpaSVAxMjM0NTY3
MRAwDgYDVQQIEWdJbMRpYw5hMQ8wDQYDVQQHEWZNaWxsZXIuIzAhBgNVBAoTGkRv
bHBoaW4gQmFuayBJbnRlcm5hdGlvbmFsMRkwFwYDVRQLExBEB2xwaGluIEJhbmsg
SU5jMR8wHQYDVQQDFBZhbGllZUBkb2xwaGluLWJhbmsuY29tMIGfMA0GCSqGSIb3
DQEBQUQUA4GNADCBiQKBgQDL+hBLxKEFU3GKquVvapkNZvs0NBFaUjntSJRhlXaW
9n0bZpmj0jtGUzfo76jZoPeMu9fV1Mw4U71CNIAx8jDdXiyAwMUzgMtS6BzFZ2/d
LTh0JDRgRqVebDQ7QueAJGaJeV/eCC47aCDqBQS3bSgxtDzLA6ShQ9TCgPKENfMl
xwIDAQAAB0AAwDQYJKoZIhvcNAQEEBQADgYEAxqp6XcuFpn+GK50XoqP0lDzp1ee2
PtpAv0Gqv5F6lmtolt/1TzV5nY29FHFjFcbGQMHNFTuKafHUykaeLo0im7+0rd9c
918FTdKKZtYU06aow4KQ67AVG2SmeT9eZ7sEd110ET0SPPw3XISZT+12J+iTiX
a+acLE40Kf2Ij/I=
-----END NEW CERTIFICATE REQUEST-----

```

Figure 11-39 Incoming request from Windows Client 2 ALICE

Once signed by the CA certificate using the following command:

```

RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.ALICE.ARM') SIGNWITH(CERTAUTH
LABEL('Dolphin Bank CA')) WITHLABEL('DB MQClient2 CA Signed')

```

the certificate looks like the one in Figure 11-40 on page 232.

```

-----BEGIN CERTIFICATE-----
MIIDHDCCAoWgAwIBAgIBCzANBgkqhkiG9w0BAQUFADBUMQswCQYDVQQGEWJVSzEY
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDVQQHEWZUZXRsZXkxEDA0BgNVBAoT
B0RvbHBoaW4xZCZAJBgNVBAsTAkhrMRkgwFgYDVQQDEW9Eb2xwaGlueJhbmGQ0Ew
HhcNMDIxMDI5MDUwMDAwWhcNMDMxMDMwMDQ1OTU5WjCBpjELMAkGA1UEBHMCMVYX
EzARBgNVBETCplJUEyMzQ1NjcxEDA0BgNVBAGTB0luZGhhbmExDzANBgNVBACT
Bk1pbGxlcjEjMCEGA1UEChMARG9scGhpbiBCYW5rIEludGVybmF0aW9uYWxGTAX
BgNVBAsTEERvbHBoaW4xZGQmFuayBJbMmXhZAdBgNVBAMUFmFsaWNLQGRvbHBoaW4t
YmFuay5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMv6EEvEoQVTcYqq
5W9qLw1m+zQ0EVPSoDNI lGGVdppb2c5tmmaM600ZTN+jvqNm94y719XUzDhTUI0
hrHyMN1eJgDaxTOAy1LoHMVnb90t0E4kNGBGpV5s5DtC54AKZol5X94ILjtoI0oF
BLdtKDGOpmSdpKFD1MKAB0Q0WZTHAgMBAAGjgZAwgY0wSwYJYIZIAYb4QgENB4T
PEdlbmVyyXRlZCBieSB0aGUgU2VjdXJlV2F5IFNlY3VyaXR5IFNlcnZlcjBm3Igt
T1MvMzkwIChSQUJGKtAdBgNVHQ4EFgQUlmyivBbly5iKJbIihpS9XVMyouoHwYD
VR0jBBgwFoAUGIUSoiU0qJL2zL9aamYBAGblIlgkWDQYJKoZIhvcNAQEFBQADgYEA
rSSsXgtjJbK4YNIWfHjAMYlPIHgECh8T/aa2jVqJA4aRvUFpYcwL5G8+/N1F7SSP
B9Mkwp69ok7b6L2yr tIHxorI4uaWC+rREc023Io3hPN/pBKeMQbrWYDq0UnPR7lg
uX+nd4dQD13ILXZhdz/B0yX9+a0kIMhZ08jTbm/Ic1=
-----END CERTIFICATE-----

```

Figure 11-40 CA signed certificate for Windows Client 2 ALICE

BOB (Windows client 3)

The incoming certificate from Windows client BOB looks like the one in Figure 11-41.

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIB5DCCAU0CAQAwwgAmxZAJBgNVBAYTALVTRMRwEQYDVQQREwpaSVAXMjM0NTY3
MRAwDgYDVQQQIEwkJmRpbW5hMQ8wDQYDVQQHEWZUZXRsZXkxIzAhBgNVBAoTGkRv
bHBoaW4xZGQmFuayBJbnRlcm5hdGlvbmFsmRkwFwYDVQQLExBEb2xwaGlueJhbmG
SW5jMRwwGgYDVQQDFBNib2JAZG9scGhpbmJhbmSuY29tMIGfMA0GCsqGSIB3DQEB
AQUAA4GNADCBiQKBgQC9yRwYpHvpBCodZzVCspMCHaxTrAwxB1waAna1DyLfbMx/
RS6ZB/IAub7FrMkdSLLwi3ZLGNhkd3s13gSf4s7TYxiiavUpPPQuN1dq1diQCwon
x75kBkHCiSdPaU91GBJUI7d1o/9DN9CJ+o67G1BtmN3g6WZ/15xDm/VoJdmNrQID
AQABoAAwDQYJKoZIhvcNAQEBBQADgYEAhIPzomaqJ/8JeV6gn5LH6tC4M/CKXCW6
+XtyG05XL1Q0yRJvSeZ01mMJ2ewueybbBsTjra/NGYoqKGI/PEq3kJfpXcUEMLB2
oJmmjDM+lcXtVT/ulwEgudhQobdB9D4Do29iIT/+HzvXrw+2aQ7GcDT/qN41gQPkY
Hhh30gm1auc=
-----END NEW CERTIFICATE REQUEST-----

```

Figure 11-41 Incoming certificate request for Windows Client 3 BOB

Once signed by the CA certificate using the following command:

```

RACDCERT ID(SDRES) GENCERT('SDRES2.CERTREQ.BOB.ARM') SIGNWITH(CERTAUTH
LABEL('Dolphin Bank CA')) WITHLABEL('DB MQClient3 CA Signed')

```

the certificate looks like the one in Figure 11-42 on page 233.

```

-----BEGIN CERTIFICATE-----
MIIDGTCCAoKgAwIBAgIBDDANBgkqhkiG9w0BAQUFADBuMQswCQYDYQQGEWJVSzEY
MBMGA1UECBMTGluY29sbnNoaXJlMQ8wDQYDYQQHEWZUZXRzZXkxEDAOBgNVBAoT
B0RvbHB0aW4xZzA1JG9wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
HhcNMDIxMDI5MDUwMDAwHhcNMDMxMDMwMDQ1OTU5WjCBZGZELMAkGA1UEBhMCVVMx
EzARBgNVBETCJlJm9udGVyZmF0aW9uY29sbnNoaXJlMQ8wDQYDYQQHEWZUZXRz
Bk1pbGxlcjEjMCEGA1UEChMARG9scGhpbiBCYW5rIEludGVybmF0aW9uY29sbnNo
BgNVBAsTEERvbHB0aW4xZzA1JG9wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
ay5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgYQAMIGJAoGBAL3JHBike+kEKh1nNUKJy
kwIdrF0sDDEHXBoCdrUPKY9szH9FLpkH8gC5vsWuYp1IsfaldmUY2GR3ezXeBJ/i
ztNjGKJq9Sk89c43V2rV2JALCifHvmQGQCkJJ09pT3UYE1Qjt3Wj/0M30In6jrsb
UG2Y3eDpZn/XnE0b9WiN2Y2tAgMBAAGjGZAwgY0wSwYJY1Z1AYb4QgENBD4TPEdl
bmVYXRlZCBieSB0aGUgU2VjdXJlV2F5IFNlY3VyaXR5IFNlcnZlcjBmb3Igt1Mv
MzkwIChSQU9uZG9wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
BBgwFoAUG1USoiU0qJL2zL9aamYBAGblIgwDQYJKoZIhvcNAQEFBQADgYEAQwTo
SadwfkWUuqT5a+HoTrrIEcddbUrICwM3MbW1HmNAwPaIue6SsqSqrHrZa7lpj6km
b30LzEQ5f77RGrToeHRCr+bKBM0q9A90qtZxxE9ZBS3DPho6zCCJD9d0EjlcQ/Av
XsAMgkd5k4vCkBGMYX/fVEBTLyLQRuGojSffkx=
-----END CERTIFICATE-----

```

Figure 11-42 CA signed certificate for Windows Client 3 BOB

Exporting the signed certificates

Once the certificate requests have been signed by the CA certificate, they need to be exported back to the requestors. Once the certificate requests have been signed by the CA certificate, they need to be exported back to the requestor using the following commands; LABEL is the one used on WITHLABEL when the CA signed it.

MQAIX2

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQAIX2 CA Signed'))
DSN('SDRES2.CERT.MQAIX2.ARM') FORMAT(CERTB64)

```

MQAIX3

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQAIX3 CA Signed'))
DSN('SDRES2.CERT.MQAIX3.ARM') FORMAT(CERTB64)

```

MQ400

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQ400 CA Signed'))
DSN('SDRES4.MQ400.CERT') FORMAT(CERTB64)

```

MQW2K2

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQW2K2 Server CA Signed'))
DSN('SDRES2.CERT.MQW2K2.ARM') FORMAT(CERTB64)

```

MQW2K5

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQW2K5 Server CA Signed'))
DSN('SDRES2.CERT.MQW2K5.ARM') FORMAT(CERTB64)

```

TOM

```

RACDCERT ID(SDRES) EXPORT(LABEL('DB MQClient1 CA Signed'))
DSN('SDRES2.CERT.TOM.ARM') FORMAT(CERTB64)

```

ALICE

```
RACDCERT ID(SDRES) EXPORT(LABEL('DB MQClient2 CA Signed'))
DSN('SDRES2.CERT.ALICE.ARM') FORMAT(CERTB64)
```

BOB

```
RACDCERT ID(SDRES) EXPORT(LABEL('DB MQClient3 CA Signed'))
DSN('SDRES2.CERT.BOB.ARM') FORMAT(CERTB64)
```

Once the certificates are exported into their export datasets, they are then FTPed back to the requestors.

Making your own certificates

We now need certificates for the z/OS systems to use, one for each channel initiator and a group certificate for the queue sharing group.

Private certificates were created for each queue manager associated with a CHINIT user ID and signed with a CA certificate for use by the clustering setup. A shared certificate could not be used because we were using individual listeners in the cluster and the user ID of each CHINIT was required in a certificate.

Important: Ensure that the Subjectsdn field for each is *unique*. We encountered problems getting the shared certificate to work and after debugging, because the field was not set up properly.

```
RACDCERT ID(MQZ1CHIN) GENCERT SUBJECTSDN(CN('MQZ1 Dolphin Bank') OU('HQ')
O('Dolphin') L('Tetley') SP('Lincolnshire') C('UK'))
withlabel('ibmWebSphereMQMQZ1') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA'))
KEYUSAGE(HANDSHAKE DATAENCRYPT)
```

```
RACDCERT ID(MQZ2CHIN) GENCERT SUBJECTSDN(CN('MQZ2 Dolphin Bank') OU('HQ')
O('Dolphin') L('Tetley') SP('Lincolnshire') C('UK'))
withlabel('ibmWebSphereMQMQZ2') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA'))
KEYUSAGE(HANDSHAKE DATAENCRYPT)
```

```
RACDCERT ID(MQZ3CHIN) GENCERT SUBJECTSDN(CN('MQZ3 Dolphin Bank') OU('HQ')
O('Dolphin') L('Tetley') SP('Lincolnshire') C('UK'))
withlabel('ibmWebSphereMQMQZ3') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA'))
KEYUSAGE(HANDSHAKE DATAENCRYPT)
```

A shared certificate was created to be used by MQ400 since it was not part of the cluster and could use the shared listener.

```
RACDCERT ID(SDRES) GENCERT SUBJECTSDN(CN('MQZG Dolphin Bank') OU('HQ')
O('Dolphin') L('Tetley') SP('Lincolnshire') C('UK'))
withlabel('ibmWebSphereMQMQZG') SIGNWITH(CERTAUTH LABEL('Dolphin Bank CA'))
KEYUSAGE(HANDSHAKE DATAENCRYPT)
```

Each of these requests produced the following error message:

```
The certificate that you are creating has an incorrect date range. The
certificate is added with NOTRUST status.
```

So we used the following commands to change the state to TRUST:

```
RACDCERT ID(MQz1CHIN) ALTER (LABEL('ibmWebSphereMQMQZ1')) TRUST
RACDCERT ID(MQZ2CHIN) ALTER (LABEL('ibmWebSphereMQMQZ2')) TRUST
RACDCERT ID(MQZ3CHIN) ALTER (LABEL('ibmWebSphereMQMQZ3')) TRUST
RACDCERT ID(SDRES) ALTER (LABEL('ibmWebSphereMQMQZG')) TRUST
```

Creating your keyring on z/OS and adding the queue manager certificates

Our initial plan was to create a shared keyring on RACF, put all the certificates in that keyring and give all the appropriate IDs authority to use it:

```
RACDCERT ID(SDRES) ADDRING(MQZGRING)
```

Note: We tried for quite a while to get this to work, but eventually found out that in order to do this, you need z/OS V1.4 or later in order to pick up the correct level of SSL support.

We were running z/OS V1.1 and V1.2, so we had to revert back to each queue manager having its own keyring. We created three keyrings, one for each queue manager, associating each one with the user ID of the queue manager's CHINIT. If all the CHINITs were running under the same user ID, one keyring would work, but in our solution we had all our CHINITs running under different user IDs to give us more granularity and control over the security on the z/OS qmgrs.

```
RACDCERT ID(MQZ1CHIN) ADDRING(MQZ1RING)
RACDCERT ID(MQZ2CHIN) ADDRING(MQZ2RING)
RACDCERT ID(MQZ3CHIN) ADDRING(MQZ3RING)
```

All the qmgrs certificates were added to each keyring using the following commands.

CA certificate

```
RACDCERT ID(MQZ1CHIN) CONNECT(CERTAUTH LABEL('Dolphin Bank CA'))  
RING(MQZ1RING) USAGE(CERTAUTH))
```

```
RACDCERT ID(MQZ2CHIN) CONNECT(CERTAUTH LABEL('Dolphin Bank CA'))  
RING(MQZ2RING) USAGE(CERTAUTH))
```

```
RACDCERT ID(MQZ3CHIN) CONNECT(CERTAUTH LABEL('Dolphin Bank CA'))  
RING(MQZ3RING) USAGE(CERTAUTH))
```

Group certificate

```
RACDCERT ID(MQZ1CHIN) CONNECT(ID(SDRES) LABEL('ibmWebSphereMQMQZG'))  
RING(MQZ1RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ2CHIN) CONNECT(ID(SDRES) LABEL('ibmWebSphereMQMQZG'))  
RING(MQZ2RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ3CHIN) CONNECT(ID(SDRES) LABEL('ibmWebSphereMQMQZG'))  
RING(MQZ3RING) USAGE(PERSONAL))
```

CHINIT certificates to MQZ1RING

```
RACDCERT ID(MQZ1CHIN) CONNECT(ID(MQZ1CHIN) LABEL('ibmWebSphereMQMQZ1'))  
RING(MQZ1RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ1CHIN) CONNECT(ID(MQZ2CHIN) LABEL('ibmWebSphereMQMQZ2'))  
RING(MQZ1RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ1CHIN) CONNECT(ID(MQZ3CHIN) LABEL('ibmWebSphereMQMQZ3'))  
RING(MQZ1RING) USAGE(PERSONAL))
```

CHINIT certificates to MQZ2RING

```
RACDCERT ID(MQZ2CHIN) CONNECT(ID(MQZ1CHIN) LABEL('ibmWebSphereMQMQZ1'))  
RING(MQZ2RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ2CHIN) CONNECT(ID(MQZ2CHIN) LABEL('ibmWebSphereMQMQZ2'))  
RING(MQZ2RING) USAGE(PERSONAL))
```

```
RACDCERT ID(MQZ2CHIN) CONNECT(ID(MQZ3CHIN) LABEL('ibmWebSphereMQMQZ3'))  
RING(MQZ2RING) USAGE(PERSONAL))
```


CHINIT certificates to MQZ3RING

```
RACDCERT ID(MQZ3CHIN) CONNECT(ID(MQZ1CHIN) LABEL('ibmWebSphereMQMQZ1'))  
RING(MQZ3RING) USAGE(PERSONAL)
```

```
RACDCERT ID(MQZ3CHIN) CONNECT(ID(MQZ2CHIN) LABEL('ibmWebSphereMQMQZ2'))  
RING(MQZ3RING) USAGE(PERSONAL)
```

```
RACDCERT ID(MQZ3CHIN) CONNECT(ID(MQZ3CHIN) LABEL('ibmWebSphereMQMQZ3'))  
RING(MQZ3RING) USAGE(PERSONAL)
```

Ensure that all user IDs needing access to the certificates have the correct authority to the appropriate RAC F profiles, using the following commands:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY)  
ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN,SDRES1,SDRES) ACCESS(UPDATE)
```

```
PERMIT IRR.DIGTCERT.LIST CLASS(FACILITY)  
ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN,SDRES1,SDRES) ACCESS(CONTROL)
```

You may well have to issue the following RACF command:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

in order for the changes to be picked up if the FACILITY class is Raclisted.

Important: We were unable to get the group certificate to work, so we had to remove it from each of the keyrings and let the shared channel use the queue manager's certificate instead. We were not sure about the cause of the failure.

Once all the channel updates and SSL task were completed, the changes required to enhance the RACF profiles and access levels needed to be made. The CLISTs found in Appendix B were used. Once these were run, we issued the following commands to refresh the security.

► RACF command

```
SETROPTS GENERIC(MQADMIN,MQQUEUE,MQNLIST,MQPROC,MQCMD) REFRESH  
SETROPTS RACLIST(MQADMIN,MQQUEUE,MQNLIST,MQPROC) REFRESH
```

These two commands will refresh the RACF dataspace with new profiles and access levels.

► MQ command

```
/=MQZ1 REFRESH SECURITY(*) CMDSCOPE(*)
```

This command will refresh the MQ view of the RACF information and perform some internal cleanup and a refresh of the information held.

11.4 WebSphere MQ setup

This section discusses the WebSphere MQ setup and the tasks performed to implement SSL in our production environment.

11.4.1 On z/OS

- ▶ Change the queue manager object on each queue manager so it has the correct KEYRING specified on the SSLKEYR attribute and specify a number of SSL tasks to be started. On WebSphere MQ for z/OS, you do this by issuing the following MQSC command on each queue manager:

```
ALTER QMGR SSLKEYR(keyringname) SSLTASKS(integer)
```

So for our qmgrs, it was :

```
=MQZ1 ALTER QMGR SSLKEYR(MQZ1RING) SSLTASKS(5)  
=MQZ2 ALTER QMGR SSLKEYR(MQZ2RING) SSLTASKS(5)  
=MQZ3 ALTER QMGR SSLKEYR(MQZ3RING) SSLTASKS(5)
```

The CHINITs need to be re-started for this to take effect.

- ▶ In order for the mover and channel definitions to use SSL, you need to make changes to your channel definitions. It is also a good idea to ensure other security attributes are set up correctly at this time. If not, change them to something more appropriate. For example, you should check the MCA USER ID and the PUTAUT attributes.

In our business scenario solution on z/OS, the following user IDs were used:

- ▶ MQZ1 qmgr runs under MQZ1MSTR
- ▶ MQZ2 qmgr runs under MQZ2MSTR
- ▶ MQZ3 qmgr runs under MQZ3MSTR
- ▶ MQZ1 CHINIT runs under MQZ1CHIN
- ▶ MQZ2 CHINIT runs under MQZ2CHIN
- ▶ MQZ3 CHINIT runs under MQZ3CHIN
- The user ID used for all administration tasks including MQ and RACF administration was SDRES1.
- User ID SDRES was used to create the shared KEYRING and owned the queue sharing group certificate.
- The MCA user ID on the clear receiver channel from MQ400 was SDRES4.

- The MCA user ID on the secure receiver channel from MQ400 was SDRES5.
- The MCA user ID on the clear cluster receiver Channel from MQAIX was SDRES2.
- The MCA user ID on the secure cluster receiver channel from MQAIX was SDRES5.
- SDRES3 was used for batch applications connecting to z/OS qmgrs.

The following commands were used to change the Channel definitions to implement SSL and additional security features:

```

CLUSTER CHANNELS:
ALTER CHANNEL(TO.CCLUS1.MQAI2) +
    CHLTYPE(CLUSSDR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP) +
    SSLCIPH(NULL_SHA +
    SSLPEER('O=Dolphin*'))
ALTER CHANNEL(TO.ECLUS1.MQAI2) +
    CHLTYPE(CLUSSDR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP) +
    SSLCIPH(TRIPLE_DES_SHA_US) +
    SSLPEER('O=Dolphin*'))

ALTER CHANNEL(TO.CCLUS1.MQZ1) +
    CHLTYPE(CLUSRCVR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP)
    MCAUSER(SDRES4) +
    SSLCAUTH(REQUIRED) +
    SSLCIPH(NULL_SHA) +
    SSLPEER('O=Dolphin*'))

DEFINE CHANNEL(TO.ECLUS1.MQZ1) +
    CHLTYPE(CLUSRCVR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP) +
    MCAUSER(SDRES5) +
    SSLCAUTH(REQUIRED) +
    SSLCIPH(TRIPLE_DES_SHA_US) +
    SSLPEER('O=Dolphin*'))

ALTER CHANNEL(TO.CCLUS1.MQZ2) +
    CHLTYPE(CLUSRCVR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP) +
  
```

```

MCAUSER(SDRES4)
SSLCAUTH(REQUIRED) +
SSLCIPH(NULL_SHA) +
SSLPEER('0=Dolphin*')

DEFINE CHANNEL(TO.ECLUS1.MQZ2) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  MCAUSER(SDRES5) +
  SSLCAUTH(REQUIRED) +
  SSLCIPH(TRIPLE_DES_SHA_US) +
  SSLPEER('0=Dolphin*')

ALTER CHANNEL(TO.CCLUS1.MQZ3) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  MCAUSER(SDRES4) +
  SSLCAUTH(REQUIRED) +
  SSLCIPH(NULL_SHA) +
  SSLPEER('0=Dolphin*')

DEFINE CHANNEL(TO.ECLUS1.MQZ3) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  MCAUSER(SDRES5) +
  SSLCAUTH(REQUIRED) +
  SSLCIPH(TRIPLE_DES_SHA_US) +
  SSLPEER('0=Dolphin*')

SENDER/RECEIVER TO MQ400
ALTER CHANNEL(MQZG.CC.MQ400) +
  CHLTYPE(SDR) +
  QSGDISP(GROUP) +
  TRPTYPE(TCP)

ALTER CHANNEL(MQZG.SC.MQ400) +
  CHLTYPE(SDR) +
  QSGDISP(GROUP) +
  TRPTYPE(TCP) +
  SSLCIPH(TRIPLE_DES_SHA_US) +
  SSLPEER('0=SHETLAND*')

ALTER CHANNEL(MQ400.CC.MQZG) +
  CHLTYPE(RCVR) +
  QSGDISP(GROUP) +
  TRPTYPE(TCP) +

```

```
MCAUSER(SDRES4)
```

```
ALTER CHANNEL(MQ400.SC.MQZG) +  
  CHLTYPE(RCVR) +  
  QSGDISP(GROUP) +  
  TRPTYPE(TCP) +  
  MCAUSER(SDRES5) +  
  SSLCIPH(TRIPLE_DES_SHA_US) +  
  SSLPEER('O=SHETLAND*') +  
  SSLCAUTH(REQUIRED)
```

Setup for the client (Windows)

- ▶ Define the SVRCONN channel. The SSLCIPH must be the same as your client definition.

```
DEFINE CHANNEL(SSL.CLCHL) +  
  CHLTYPE(SVRCONN) TRPTYPE(TCP) +  
  SSLPEER('O=Dolphin') +  
  SSLCIPH(TRIPLE_DES_SHA_US)  
  SSLAUTH(REQUIRED)
```

- ▶ Define a CLNTCONN channel when you use the channel table file for your WebSphere MQ client. The SSLCIPH must be the same as your server definition.

```
DEFINE CHANNEL(SSL.CLCHL) +  
  CHLTYPE(CLNTCONN) TRPTYPE(TCP) +  
  SSLPEER('O=Dolphin') +  
  SSLCIPH(TRIPLE_DES_SHA_US)
```

Client (Windows)

- ▶ Channel table file

If you use a channel table file, copy AMQCHLCL.TAB from your WebSphere MQ server.

Set MQSSLKEYR to your key store file name:

```
export MQSSLKEYR=c:\ssl:mykey
```

Run your application.

- ▶ MQCONN sample program

Set your keystore file name to MQCS0.KeyRepository and CipherSpec to MQCD.SSLCipherSpec.

This sample program reads an input file called infile for the put message.

It connects the Queue manager specified to the parameter by a SVRCONN channel named SSL.CLCHL:

```
sslput QName QmgrName ConName Message-Count
```



```
Command Prompt
C:\mayuni>
C:\mayuni>sslputc LQ1 MQW2K2 localhost 2
Channel Name      :SSL.CLCHL
Key Repository:C:\SSL\mykey
CipherSpec       :TRIPLE_DES_SHA_US

Opening the queue LQ1

*   T   h   i   s   i   s   i   s   a   t   e   s   t   m   e   s   s   a   g   e   .

2 PUT successful!! Disconnecting from QMgr...
C:\mayuni>
```

Figure 11-43 A program 'sslput' execution image

```
#include <stdio.h>
#include <string.h>
/*****
/* Samle Program for MQ Client with SSL (sslput.c) */
/*****
/* "sslput" write the messages in the "inpfile". */
/* */
/*****
#include <stdlib.h>
#include <cmqc.h>
#include <cmqxc.h>

main(int argc, char *argv[]){
    FILE *fp;
    char buffer[100];
    MQLONG i=0;
    MQOD mqod = {MQOD_DEFAULT};
    MQMD mqmd = {MQMD_DEFAULT};
    MQPMO pmo = {MQPMO_DEFAULT};
    MQCNO connect_options = {MQCNO_DEFAULT};
    MQCD mycd = {MQCD_CLIENT_CONN_DEFAULT};
    MQSCO mysco = {MQSCO_DEFAULT};
    MQCHAR qmname[MQ_Q_MGR_NAME_LENGTH];
    MQHCONN Hcon;
    MQHOBJ Hobj;
    MQLONG open_options;
    MQLONG close_options;
    MQLONG compcode;
    MQLONG reason;
    int cc;
```

```

int count;

if(argc!=5){
    printf("Usage: sslputc QName QmName ConnName Message-Count");
    exit(1);
}

count = atoi(argv[4]);
strncpy(qmname, argv[2], MQ_Q_MGR_NAME_LENGTH);
strncpy(mycd.ConnectionName, argv[3], MQ_CONN_NAME_LENGTH);
strncpy(mycd.ChannelName, "SSL.CLCHL", MQ_CHANNEL_NAME_LENGTH);
printf("Channel Name  :%s\n", mycd.ChannelName);

/* For SSL */
/** Specify Keystore file name without ".sto" */
strncpy(mysco.KeyRepository, "C:\\SSL\\mykey",
MQ_SSL_KEY_REPOSITORY_LENGTH);
printf("Key Repository:%s\n", mysco.KeyRepository);

/** Specify CipherSpec */
strcpy(mycd.SSLCipherSpec,"TRIPLE_DES_SHA_US");
connect_options.SSLConfigPtr = &mysco;
connect_options.ClientConnPtr=&mycd;
connect_options.Version = MQCNO_VERSION_4;
printf("CipherSpec   :%s\n", mycd.SSLCipherSpec);

/* Connect Qmgr */
MQCONN(qmname, &connect_options, &Hcon, &compcode, &reason);
if (compcode == MQCC_FAILED){
    printf("MQCONN ended with reason code %ld\n", reason);
    exit( (int)reason );
}
strcpy(mqod.ObjectName, argv[1]);

/* Open a Queue */
open_options = MQOO_OUTPUT + MQOO_FAIL_IF QUIESCING;
printf("\nOpening the queue %s\n", mqod.ObjectName);
MQOPEN(Hcon, &mqod, open_options, &Hobj, &compcode, &reason);
if(reason!=MQRC_NONE){
    printf("MQOPEN failed with reason code %ld\n", reason);
    exit((int)reason);
}

if (compcode == MQCC_FAILED){
    printf("unable to open queue for output\n");
}

/* Open the "inpfile" and get messages */
if((fp=fopen("inpfile", "r"))==NULL){

```

```

        printf("Cannot read file: inpfiler\n");
        exit(1);
    }
    while((buffer[i]=fgetc(fp))!=EOF){
        printf("%c ",buffer[i]);
        i++;
    }

/* Set MQMD */
    memcpy(mqmd.Format,MQFMT_STRING, (unsigned int)MQ_FORMAT_LENGTH);
    mqmd.Persistence = MQPER_PERSISTENT;
    memcpy(mqmd.MsgId, MQMI_NONE, sizeof(mqmd.MsgId));
    strcpy(mqmd.ReplyToQ, "REPORTQ");

/* Put the message */
    for(cc=0; cc<count; cc++){
        MQPUT(Hcon, Hobj, &mqmd, &pmo, i, buffer, &compcode, &reason);
        if (reason != MQRC_NONE){
            printf("MQPUT ended with reason code %ld\n", reason);
        }
    }
    fclose(fp);

/* Close the queue */
    close_options = 0 ;
    MQCLOSE(Hcon, &Hobj, close_options, &compcode, &reason);
    printf("\n%d PUT successful!! Disconnecting from QMgr...", count);

/* Disconnect QMgr */
    MQDISC(&Hcon, &compcode, &reason);

}

```

Verifying the SSL client

A packet sniffer was started with a filter to capture data destined for port 1414. The sniffer collected all of the packets and their contents from a WebSphere MQ client put.


```

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>SET MQSERVER=CLIENT.TEST/TCP/ITSOAIX2

C:\>AMQSPUTC LQ1 MQAIX2
Sample AMQSPUT0 start
target queue is LQ1
TESTMSG

Sample AMQSPUT0 end

C:\>_

```

Figure 11-44 Commands issued and message sent

```

54 53 48 20 00 00 00 84 02 01 01 00 00 00 00 00 TSH .....
00 00 00 00 22 02 00 00 52 03 00 00 49 44 20 20 .....R...ID
07 25 00 00 00 00 32 00 FE 7F 00 00 00 00 40 00 .%.2.....@.
FF C9 9A 3B 43 4C 49 45 4E 54 2E 54 45 53 54 20 ...;CLIENT.TEST
20 20 20 20 20 20 20 20 56 00 52 03 20 20 20 20 V.R.
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00 00 73 20 .....
..s

```

Figure 11-45 Call to channel

```

54 53 48 20 00 00 00 A0 02 08 01 00 00 00 00 00 TSH .....
00 00 00 00 11 01 00 00 52 03 00 00 55 49 44 20 .....R...UID
53 43 41 4E 4C 41 4E 20 20 20 20 20 20 20 20 20 SCANLAN
20 20 20 20 20 20 20 20 73 63 61 6E 6C 61 6E 20 scanlan
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
00 05 15 00 00 00 78 2E 9D 13 23 5F 63 6B 07 E5 .....x...#_ck..
3B 2B EC 03 00 00 00 00 00 00 00 00 00 00 00 00 ;+.....

```

Figure 11-46 User ID

```

54 53 48 20 00 00 00 A4 02 81 30 00 00 00 00 00 TSH .....0....
00 00 00 00 11 01 00 00 52 03 00 00 00 00 00 A4 .....R.....
00 00 00 00 00 00 00 00 00 00 00 00 00 4D 51 41 49 .....MQAI
58 32 20 20 20 20 20 20 20 20 20 20 20 20 20 20 X2
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 65 62 53 70 ebSp
68 65 72 65 20 4D 51 5C 62 69 6E 5C 61 6D 71 73 here MQ\bin\amqs
70 75 74 63 2E 65 78 65 0B 00 00 00 16 01 05 15 putc.exe.....
00 00 00 78 2E 9D 13 23 5F 63 6B 07 E5 3B 2B EC ...x...#_ck..;+
03 00 00 00 00 00 00 00 00 00 00 00 0B 01 00 00 00 .....
00 00 00 00 .....

```

Figure 11-47 Queue manager name and program

```

54 53 48 20 00 00 00 00 D8 02 83 30 00 00 00 00 00 TSH .....0.....
00 00 00 00 11 01 00 00 52 03 00 00 00 00 00 00 D8 .....R.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 4F 44 20 20 .....0D
01 00 00 00 01 00 00 00 4C 51 31 00 00 00 00 00 .....LQ1.....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 41 4D 51 2E 2A 00 00 00 .....AMQ.*...
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

```

Figure 11-48 Queue name

```

00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
50 4D 4F 20 01 00 00 00 00 00 00 00 00 00 FF FF FF FF PHO .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
07 00 00 00 54 45 53 54 4D 53 47 .....TESTMSG

```

Figure 11-49 Message data

The process was repeated using SSL encryption on the SVRCONN/CLNTCONN pair.

```

Microsoft Windows 2000 [Version 5.00.2195]
<C> Copyright 1985-2000 Microsoft Corp.

C:\>SET MQSSLKEYR=C:\KEY

C:\>SET MQCHLLIB=C:\

C:\>AMQSPUTC LQ1 MQAIX3
Sample AMQSPUT0 start
target queue is LQ1
TESTMSG

Sample AMQSPUT0 end

C:\>

```

Figure 11-50 Commands and message sent

```

16 03 00 00 2F 01 00 00 2B 03 00 FB 9E 1F CA 01 ..../.+......
36 18 01 8F 1E 35 67 A4 28 4D 8D FA 7D C9 8C 96 6....5g.(M...)...
8F DB EF 1A 97 EC 72 C4 F9 EF 9C 00 00 04 00 0A .....r.....
00 13 01 00 .....

```

Figure 11-51 The call to the channel

```

16 03 00 03 BF 0B 00 02 E1 00 02 AE 00 02 AB 30 .....0
82 02 A7 30 82 02 10 A0 03 02 01 02 02 01 03 30 ...0.....0
0D 06 09 2A 86 48 86 F7 0D 01 01 05 05 00 30 6E ...*.H.....0m
31 0B 30 09 06 03 55 04 06 13 02 55 4B 31 15 30 1.0...U...UK1.0
13 06 03 55 04 08 13 0C 4C 69 6E 63 6F 6C 6E 73 ...U...Lincolns
68 69 72 65 31 0F 30 0D 06 03 55 04 07 13 06 54 hire1.0...U...T
65 74 6C 65 79 31 10 30 0E 06 03 55 04 0A 13 07 etley1.0...U...
44 6F 6C 70 68 69 6E 31 0B 30 09 06 03 55 04 0B Dolphin1.0...U..
13 02 48 51 31 18 30 16 06 03 55 04 03 13 0F 44 ..Hq1.0...U...D
6F 6C 70 68 69 6E 20 42 61 6E 6B 20 43 41 30 1E olphin Bank CA0.
17 0D 30 32 31 30 32 33 30 34 30 30 30 30 5A 17 ..0210230400002.

```

Figure 11-52 Certificate exchange

```

17 03 00 02 10 0B 82 DA DC 31 C2 C1 86 AD 2B AD .....l....+.
9F A7 5E 2E 1D D8 ED CB 5C A6 06 85 84 DE C8 FE ..^.....\.....
82 ED 2E 75 5D 72 32 98 E4 E1 40 EE 1F 97 1E 58 ..+u|r2...@....X
D6 D7 E1 2A 66 01 56 3E 11 6F 67 54 B6 42 73 B4 ...*f.V>.ogT.Bs.
E1 93 79 90 01 C3 87 50 68 71 D0 FC 22 CF 72 9A ..y....Phq..."r.
A3 80 35 90 5C B0 B8 19 3E 75 C0 B8 57 D2 A9 7E ..S.\...>u..W...~
37 E2 F2 A2 15 37 56 43 05 4B CB 78 E6 43 56 2F 7....7VC.K.x.CV/
60 70 34 6D 9E 87 05 72 84 86 09 2F F0 A6 A7 44 `p4m....r.../...D
EE E6 87 66 01 4B B8 15 4B 45 5E 16 6F FE 05 97 ...f.K..KE^..o...
3B 97 3A 03 98 6B 1F 51 02 C4 F7 23 E6 B4 47 91 ;...k.Q...#.G.
73 0F EE F5 37 C4 1E 53 8E 41 34 7B DA CB 01 08 s...7..S.A4{....

```

Figure 11-53 Encrypted messages

Checking the channel with SSL

You can see the DN of the other side channel in the SSLPEER value in the channel status, not the channel definition.

```

aixterm
:
dis chs(TO.ECLUS1.MQAI3) all
  14 : dis chs(TO.ECLUS1.MQAI3) all
AMQ8417: Display Channel Status details.
CHANNEL(TO.ECLUS1.MQAI3)          XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
CONNAME(9.20.18.175(1414))        CURRENT
CHLTYPE(CLUSSDR)                 INDOUBT(NO)
LSTSEQNO(236)                    LSTLUWID(3DC7B37D10000101)
CURMSGS(0)                       CURSEQNO(236)
CURLUWID(3DC7B37D10000102)       STATUS(RUNNING)
LSTMSGTI(12.03.15)              LSTMSGDA(2002-11-05)
MSGs(1)                          BYTSSENT(3572)
BYTSRCVD(188)                   BATCHES(2)
BATCHSZ(50)                     HBINT(300)
NPMSPEED(FAST)                  CHSTATI(12.03.09)
CHSTADA(2002-11-05)             BUFSSENT(3)
BUFSRCVD(3)                     LONGRTS(999999999)
SHORTRTS(10)                    JOBNAME(00002BD800000006)
MCASTAT(RUNNING)                STOPREQ(NO)
LOCLADDR(9.20.18.71(34417))
SSLPEER(CN=MQAI3,OU=Tetley,0=Dolphin Bank,L=Tetley,ST=Lincolnshire,C=GB)
RQMNAME(MQAI3)

```

Figure 11-54 Display channel status on MQAI2

```

aixterm
:
dis chs(TO.ECLUS1.MQAI2) all
  1 : dis chs(TO.ECLUS1.MQAI2) all
AMQ8417: Display Channel Status details.
CHANNEL(TO.ECLUS1.MQAI2)          XMITQ(SYSTEM.CLUSTER.TRANSMIT.QUEUE)
CONNAME(9.20.18.71(1414))        CURRENT
CHLTYPE(CLUSSDR)                 INDOUBT(NO)
LSTSEQNO(851)                    LSTLUWID(3DC7B52510000001)
CURMSGS(0)                       CURSEQNO(851)
CURLUWID(3DC7B52510000002)       STATUS(RUNNING)
LSTMSGTI(12.10.13)              LSTMSGDA(2002-11-05)
MSGs(5)                          BYTSSENT(14348)
BYTSRCVD(160)                   BATCHES(1)
BATCHSZ(50)                     HBINT(300)
NPMSPEED(FAST)                  CHSTATI(12.10.11)
CHSTADA(2002-11-05)             BUFSSENT(6)
BUFSRCVD(2)                     LONGRTS(999999999)
SHORTRTS(10)                    JOBNAME(00005A3200000018)
MCASTAT(RUNNING)                STOPREQ(NO)
LOCLADDR(9.20.18.175(32845))
SSLPEER(CN=MQAI2,OU=Sapporo,0=Dolphin Bank,L=Sapporo,ST=Hokkaido,C=JP)
RQMNAME(MQAI2)

```

Figure 11-55 Display channel status on MQAI3

11.4.2 On OS/400

On WebSphere MQ for OS/400, you must do the following.

HTTP server

A Digital Certificate Manager (DCM) is needed. The setup in the WebSphere MQ Security manual provides clear guidance on how to set up the MQ components once the AS400 HTTP administration server is running.

To get the HTTP server running, make sure you have these software pre-requisites:

- ▶ OS/400 Version 5 Release 2 (5722-SS1)
- ▶ IBM Developer Kit for Java (5722-JV1) *BASE, Option 3, and Option 4
- ▶ Crypto Access Provider 128-bit for iSeries (5722-AC3)
- ▶ OS/400 Digital Certificate Manager option (5722-SS1) Option 34

For further information, consult:

<http://publib.boulder.ibm.com/series/v5r2/ic2924/index.htm?info/rzaie/rzaieinstalling.htm>

Issue the following commands:

```
strtcpsvr *http *admin
wrkactjob to qttpsvr (ask administrator for status codes)
wrkjob admin 1,4,5 display job logs
chgtcpdmn to add hostname from dns to machine ??
addtcphte ( relate IP address to both itso400 and itso400.hursley.ibm.com)
netstat (look to see as_admin is listening on port 2001, 'F14 to toggle)
```

Digital certificate manager (DCM)

In the Web browser <http://servername:2001>, log on with a user profile with sufficient authority and then follow the **DCM** link.

IBM®
(C) IBM Corporation 2000

AS/400 Tasks

ITSO400.HURSLEY.IBM.COM

- IBM HTTP Server for AS/400**
Configure the AS/400 HTTP Server and SSL
- Digital Certificate Manager**
Create, distribute, and manage Digital Certificates
- IBM IPP Server for AS/400**
Configure the IBM IPP Server
- 4758 Cryptographic Coprocessor**
Configure the 4758 coprocessor

Figure 11-56 HTTP administration

If you wish to use OS/400 as your CA, follow the link **Create a Certificate Authority**.

Digital Certificate Manager

Create a Certificate Authority (CA)

Certificate type: Certificate Authority (CA)
Certificate store: Local Certificate Authority (CA)

The system will create a certificate with a private key and store the certificate in the Local Certificate Authority (CA) certificate store.

Key size: (bits)

Certificate store password: (required)

Confirm password: (required)

Certificate Information

Certificate Authority (CA) name: (required)

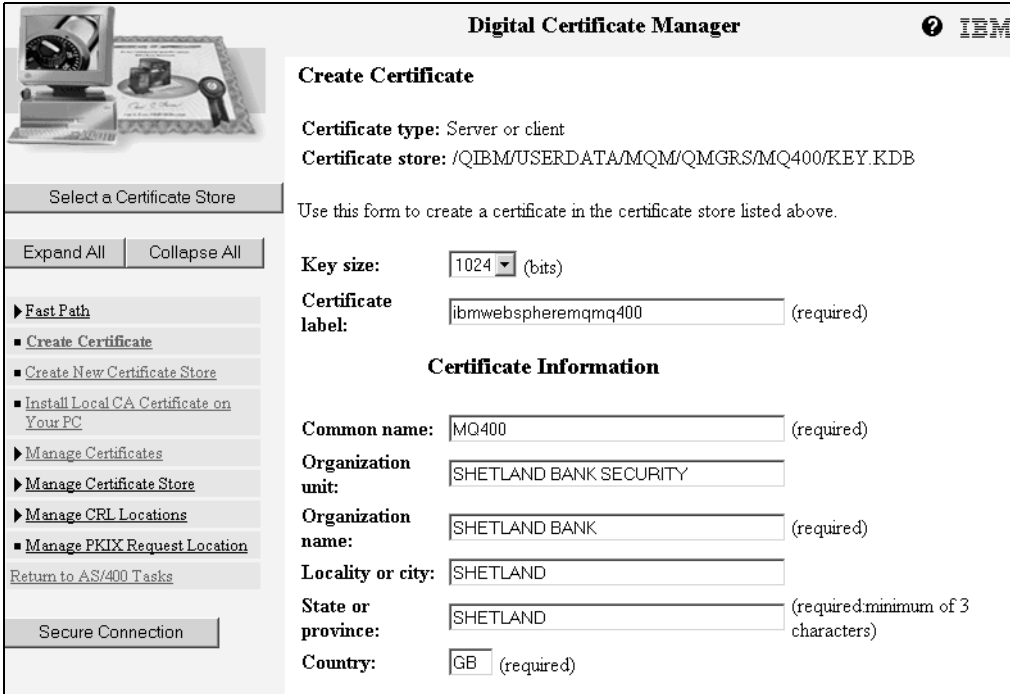
Organization unit:

Left sidebar navigation:
 Select a Certificate Store
 Expand All | Collapse All
 Manage User Certificates
 Create New Certificate Store
 Create a Certificate Authority (CA)
 Manage CRL Locations
 Manage PKIX Request Location
 Return to AS/400 Tasks
 Secure Connection

Figure 11-57 Digital certificate manager

In our business scenario, RACF was the CA, so we needed to create a certificate store for the queue manager, assign it to the queue manager and generate a key

pair for signing. We followed the steps in the *WebSphere MQ Security* manual. (see the sections *Setting up a key repository* followed by *Obtaining personal certificates*).



Digital Certificate Manager IBM

Create Certificate

Certificate type: Server or client
Certificate store: /QIBM/USERDATA/MQM/QMGRS/MQ400/KEY.KDB

Select a Certificate Store

Use this form to create a certificate in the certificate store listed above.

Expand All Collapse All

Fast Path

- [Create Certificate](#)
- [Create New Certificate Store](#)
- [Install Local CA Certificate on Your PC](#)
- ▶ [Manage Certificates](#)
- ▶ [Manage Certificate Store](#)
- ▶ [Manage CRL Locations](#)
- [Manage PKIX Request Location](#)

[Return to AS/400 Tasks](#)

Secure Connection

Key size: 1024 (bits)

Certificate label: (required)

Certificate Information

Common name: (required)

Organization unit:

Organization name: (required)

Locality or city:

State or province: (required: minimum of 3 characters)

Country: (required)

Figure 11-58 Create certificate

Note: The label must be `ibmwebspheremqmgr_name`.

Import the CA certificate from Fast Path, click **Work with CA certificates** then select **Import**.



Figure 11-59 Import CA certificate

Select the queue manager certificate store.

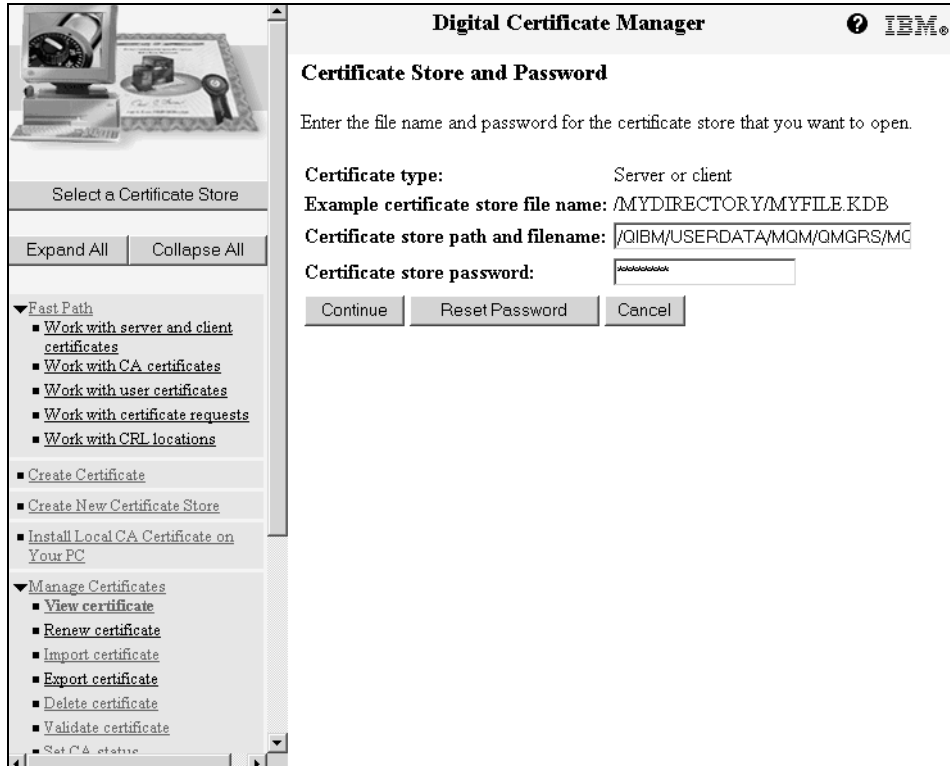


Figure 11-60 Import user certificate

Import the certificate in Manage Certificates.

In the TN5250 section, enter the command WRKMQM and select option **2**.

Enter the value for SSLKEYR AND SSLKEYRPWD (the same as was allocated previously when creating the keystore).

```

Change Message Queue Manager (CHGMQM)

Type choices, press Enter.
SSL CRL Namelist . . . . . *NONE
SSL Key Repository . . . . . '/QIBM/UserData/mqm/qmgrs/MQ400/key'
SSL Repository Password . . . . . 12345678

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 11-61 Change queue manager

Permit qmqmadm to key.kdb.

```

Change Authority (CHGAUT)

Type choices, press Enter.
Object . . . . . > '/qibm/userdata/mqm/qmgrs/mq400/KEY.KDB'
User . . . . . > QMQMADM      Name, *PUBLIC, *NTWIRF
New data authorities . . . . . > *RWX      *SAME, *NONE, *RWX, *RX...
New object authorities . . . . . > *ALL      *SAME, *NONE, *ALL...
+ for more values
Authorization list . . . . . _____ Name, *NONE

F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F13=How to use this display
F24=More keys
Bottom

```

Figure 11-62 Change object authority

Set SSL CipherSpec on sender and receiver channels at both ends.

```

Change MQ Channel (CHGMQCHL)

Type choices, press Enter.

Message exit user data . . . . . *SAME
+ for more values
Convert message . . . . . *SAME *YES, *NO, *SAME
Sequence number wrap . . . . . *SAME 100-999999999, *SAME
Maximum message length . . . . . *SAME 0-104857600, *SAME
Heartbeat interval . . . . . *SAME 0-999999999, *SAME
Non Persistent Message Speed . . *SAME *FAST, *NORMAL, *SAME
SSL Client Authentication . . . . > *REQUIRED *REQUIRED, *OPTIONAL, *SAME
SSL CipherSpec . . . . . > *TRIPLE_DES_SHA_US
SSL Peer name . . . . . *SAME

. . . More...

F3=Exit F4=Prompt F5=Refresh F12=Cancel F13=How to use this display
F24=More keys

```

Figure 11-63 Change MQ channel

11.4.3 On AIX

This sample shows only AIX, but the definition is the same on all platforms.

- ▶ Set your key repository name to the queue manager's attribute SSLKEY.

```
ALTER QMGR SSLKEYR('/var/mqm/qmgrs/MQAI2/ssl/key')
```

- ▶ Alter the cluster receiver channel's attribute for SSL. You have to specify SSLCIPH. This must be the same as for all cluster channels, and SSLPEER must be set your filtering

```
ALTER CHANNEL(TO.ECLUS1.TO.MQAI2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(TRIPLE_DES_SHA_US)
```

```
ALTER CHANNEL(TO.ECLUS2.TO.MQAI2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(TRIPLE_DES_SHA_US)
```

```
ALTER CHANNEL(TO.CCLUS1.TO.MQAI2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(NULL_SHA)
```

```
ALTER CHANNEL(TO.CCLUS2.TO.MQAI2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(NULL_SHA)
```

- ▶ When migrating from a non-SSL cluster to an SSL cluster, you must change each CLUSRCVR and ensure that the change is fully reflected around the cluster before changing the next one. Each change will be delivered along consistent channels to all the places it should go. If you change many at once, the clustering software will probably attempt to flow a change along another channel on which the CLUSRCVR change has not yet been reflected to the auto-defined cluster sender. This channel will not start because, at this instant, SSL is only defined on the CLUSRCVR. So if you change all the CLUSRCVRs at once, the updates will probably not flow around fully and the resulting cluster will not work properly.
- ▶ Alter the definition of the cluster sender channel to the full repository that you define explicitly so it has the correct SSL parameters. You must do this because otherwise, the **REFRESH** command with the **REPOS(YES)** option will not work.

```
ALTER CHANNEL(TO.ECLUS1.TO.MQAIX3) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(TRIPLE_DES_SHA_US)
```

```
ALTER CHANNEL(TO.ECLUS2.TO.MQAIX3) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(TRIPLE_DES_SHA_US)
```

```
ALTER CHANNEL(TO.CCLUS1.TO.MQAIX3) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(NULL_SHA)
```

```
ALTER CHANNEL(TO.CCLUS2.TO.MQAIX3) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  SSLPEER('0=Dolphin') +
  SSLCIPH(NULL_SHA)
```

Important: You only change the CLURCVR channel definition and restart the channel from the other side. Once you change the CLUSRCVR definition, it is delivered to all the auto-defined CLUSSDR channels when it is restarted.

11.4.4 On Windows

On WebSphere MQ for Windows 2000, you should do the following.

Clients

Client channels are authenticated using SSLPEER settings on the channels. MCAUSER is still set to mquser1. However, we limit access to the channel by checking the distinguished name of the client. The information is enciphered for all traffic. The setup has already been discussed.

OAM

In order to lock down the queues that a client connection over a secured channel can put to in the cluster, we created alias queues:

```
DEFINE QALIAS ('ACCOUNT.REQUEST.EC.LOCAL') +
  TARGQ('ACCOUNT.REQUEST.EC') +
  REPLACE
```

```
DEFINE QALIAS ('ATM.REQUEST.CC.LOCAL') +
  TARGQ('ATM.REQUEST.CC') +
  REPLACE
```

```
DEFINE QALIAS ('PORTFOLIO.REQUEST.EC.LOCAL') +
  TARGQ('PORTFOLIO.REQUEST.EC') +
  REPLACE
```

The OAM was changed to deny global access to queues in a cluster, and limit the access to these queues only. Therefore, this limits the queues that a client can put to. The OAM script follows:

```
@echo off
set QMGR=MQW2K5

REM - REMOVE PERMISSIONS PREVIOUSLY SET -
setmqaut -m %QMGR% -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t q -g MQAPPS -all

REM -- Give rights to connect to the queue manager ----
setmqaut -m %QMGR% -t qmgr -g MQAPPS +connect

REM - Give rights to get from response queues -
setmqaut -m %QMGR% -n **.RESPONSE.* -t q -g MQAPPS +get +browse

REM - Give rights to new local objects -
setmqaut -m %QMGR% -n **.REQUEST.* -t q -g MQAPPS +put
```

Channels

All E class channels were secured using SSL. The steps to complete this have already been covered and the scripts are given in Appendix B.

11.5 Cryptography: US export regulations

A good description of US cryptography export regulations and their relaxation can be found at the following URL:

<http://www.law.indiana.edu/fclj/pubs/v53/no2/black1.pdf>

This site contains an article from the Federal Communications Law Journal entitled *Taking Account of the World As It Will Be: The Shifting Course of U.S. Encryption Policy*.

The article is surprisingly readable, though it tells you much more than you may want to know.



A

Good security practices

There are a number of security exposures that seem to be common to many WebSphere MQ customer installations. These often stem from default options that are accepted by customers during WebSphere MQ installation and seldom changed or reviewed.

This chapter discusses good security practices which can help avoid some of the common exposures by organizations.

Naming conventions

Think about your queue naming conventions on z/OS to simplify RACF profiles. For example, use some sort of “company.dept.application” hierarchy, so you can assign access to “company.*”, “company.dept.*” or “company.dept.application.*”. Avoid common mistakes, such as putting the queue type at the start of the queue name, for example, queue local (QL) or queue remote (QR): QL.MYLOCALQ, QR.MYREMOTEQ, and so on.

Assign rights

Be aware that assigning rights to a "principal" under UNIX assigns those rights to the principal's primary group. As a default, the UNIX user's primary group is set to `staff`, so you should create a new group for WebSphere MQ application and add your user to it.

OAM

If you use MQSeries V5.1, when backing up your queue manager, make sure you back up the OAM files as well. OAM files can be restored simply by putting them in the right location in the directory tree. There is then no need to run a script of `setmqaut` commands.

Important: From MQSeries V5.2 onwards, OAM uses the queue called `SYSTEM.AUTH.DATA.QUEUE` to store authorization data instead of files. Therefore, you do not have to back up the OAM directory.

Delete default items

The WebSphere MQ installation process places many default objects on a system that can later on allow attacks on the infrastructure. These should all be deleted or protected prior to moving to a production environment. Objects that start with `SYSTEM.DEFAULT` should be carefully examined as to their usage. If you do not know why an object is there, rename it and see if the system requires having it back; if not, delete it.

Protect command server queues

Command server queues such as SYSTEM.COMMAND.INPUT on z/OS and SYSTEM.ADMIN.COMMAND.QUEUE on UNIX and Windows are gateways into a WebSphere MQ infrastructure that should be protected as much as any other major electronic gateway into an enterprise.

By default, the command server is not started in the UNIX environment. In the Windows environment, if you start a queue manager from Windows Explorer, the command server is started by default (since this is how the Windows Explorer communicates with the queue manager). However, if you start a queue manager from the DOS command line on Windows, the command server is not started by default.

Do not place everyone into the mqm group

Placing everyone who wants to use tools such as MQ Explorer into the mqm group is a bad mistake. You should create a new group with access rights similar to those granted to mqm and then create new groups for differing levels of users, such as mqadmin, mqview, mqops and so on. Respective users should be placed in those groups.

Never have a blank MCAUSER

When defining channels, never accept the default “blank” for MCAUSER. Channels are a major electronic highway into an enterprise and they should be protected as such.

Change the CMDUSER on z/OS

The default user ID for command security checks is CSQOPR. It is recommended that you change this (in CSQ6SYSP) and give restricted authority to the ID.

Alias the DLQ

Aliasing the “read” Dead Letter Queue (DLQ) allows the enterprise to provide browse access to many and write access to few. Any reasonably organized infrastructure attack on a WebSphere MQ environment would attempt to obscure

actions through DLQ and log file cleanup. Restricting such access will allow security staff to perform forensic analysis.

ALTUSER ID

Use of an alternate user ID in the MQMD should be carefully examined since it potentially allows a prospective attacker to subvert object authority.

Blank user IDs

RACF allows user IDs to be blank. Access to these user IDs should be carefully controlled.

The default security time-out value on z/OS is large. Customers should consider setting this to a shorter time interval to minimize the security risk. The value is maintained in the queue manager object.

Environment variables

Systems should be automatically checked for the presence of the MQSNOAUT=YES environment variable on UNIX and Windows platforms. Setting this variable disables the OAM, but does not disable other security functions.

Linear logging

Many customers define linear logging for UNIX and Windows queue managers for the sake of expediency (a nice way of saying that it is too complex to worry about backing up linear logs). This will prove to be an expensive choice when the enterprise needs to recover at a point in time because of a security breach.

Tip: There is a supportpac Linear log clean-up script. This SupportPac provides a Perl script that can be used to either compress, gzip or delete linear logfiles that are no longer needed by MQSeries. Please refer to:

<http://www-3.ibm.com/software/ts/mqseries/txppacs/ms62.html>

Generic profiles

Defining too many generic profiles to RACF will prove to be costly in terms of system maintenance, as will not setting up RESLEVEL correctly. Customers should carefully plan out what resources they need to protect, and how they will protect those resources, before touching a single RACF definition.

ISPF panel access

Access to the ISPF panels available for WebSphere MQ for z/OS should be controlled to ensure that objects cannot be defined, altered, started or stopped without adequate authority.

LU6.2 flowed user ID

Many customers attempt to flow user IDs through an APPC channel without correctly defining APPC security, for example for the APPCLU RACF class. The *z/OS System Setup Guide* explains how to correctly perform this task in the section entitled "Channel Initiator Security".

Use aliases

Aliases can be extremely useful if you want to set different permissions on a queue for read and write access without having to create many different profiles or permissions.

CHINIT user IDs

CHINITs should run under different user IDs in a queue sharing group to enable different access profiles to be applied to differing access paths.



Scripts, samples code and JCL

This appendix contains the following:

- ▶ Scripts, which show:
 - The object definitions used to set up the business scenario environment.
 - CLISTS used to create RACF definitions for the Queue Sharing Group(QSG) MQZG on z/OS.
- ▶ JCL
Examples of the JCL we used throughout the project.
- ▶ Code
Code samples we used to test our business scenario scenarios.

This appendix is split into two sections, “Scripts” and “JCL and Code”. The latter also contains various program sources used in the setup for our business scenario. In each section, the information provided is listed by platform.

Script files by platform

This section contains script files our business scenario setup listed by platform.

z/OS

Listed below are script files for z/OS.

Object definitions

Example: B-1 The following scripts show the definitions used to set up the objects in the QSG MQZG

```
*z/OS object definitions (INITIAL SETUP)
*-----
*IF A SPECIFIC OPTION IS NOT SPECIFIED WE PICK UP THE DEFAULTS

*z/OS to MQ400 object definitions
*-----
*SHARED QUEUES:
*
DEFINE QLOCAL(PORTFOLIO.BATCH.COMPLETE.SHARED)  +
      CFSTRUCT(APPLICATION1)  +
      QSGDISP(SHARED)  +
      DEFSOPTS(SHARED)  +
      SHARE
DEFINE QLOCAL(PORTFOLIO.BATCH.COMPLETE.SHARED.SC)  +
      CFSTRUCT(APPLICATION1)  +
      QSGDISP(SHARED)  +
      DEFSOPTS(SHARED)  +
      SHARE

*REMOTE QUEUES:
*
DEFINE QREMOTE(PORTFOLIO.BATCH.MQ400)  +
      QSGDISP(GROUP)  +
      RNAME(PORTFOLIO.BATCH)  +
      RQMNAME(MQ400)  +
      XMITQ(MQ400)
DEFINE QREMOTE(PORTFOLIO.BATCH.SC.MQ400)  +
      QSGDISP(GROUP)  +
      RNAME(PORTFOLIO.BATCH.SC)  +
      RQMNAME(MQ400)  +
      XMITQ(MQ400.SC)

*TRANSMISSION QUEUES:
```

```

*
DEFINE QLOCAL(MQ400) +
    QSGDISP(SHARED) +
    USAGE(XMITQ)
DEFINE QLOCAL(MQ400.SC) +
    QSGDISP(SHARED) +
    USAGE(XMITQ)

*CHANNELS:
*
DEFINE CHANNEL(MQZG.CC.MQ400) +
    CHLTYPE(SDR) +
    QSGDISP(GROUP) +
    TRPTYPE(TCP) +
    CONNAME('9.20.19.212') +
    XMITQ(MQ400)
DEFINE CHANNEL(MQZG.SC.MQ400) +
    CHLTYPE(SDR) +
    QSGDISP(GROUP) +
    TRPTYPE(TCP) +
    CONNAME('9.20.19.212') +
    XMITQ(MQ400.SC)
DEFINE CHANNEL(MQ400.CC.MQZG) +
    CHLTYPE(RCVR) +
    QSGDISP(GROUP)
DEFINE CHANNEL(MQ400.SC.MQZG) +
    CHLTYPE(RCVR) +
    QSGDISP(GROUP)

*z/OS to MQAIX object definitions (INCLUDES CLUSTER CHANNELS)
*-----
*SHARED QUEUES:
*
DEFINE QLOCAL(ACCOUNT.REQUEST.EC.CICS) +
    CFSTRUCT(APPLICATION1) +
    QSGDISP(SHARED) +
    DEFSOPTS(SHARED) +
    CLUSTER(ECLUS1) +
    SHARE
DEFINE QLOCAL(PORTFOLIO.REQUEST.EC.CICS) +
    CFSTRUCT(APPLICATION1) +
    QSGDISP(SHARED) +
    DEFSOPTS(SHARED) +
    CLUSTER(ECLUS1) +
    SHARE
DEFINE QLOCAL(ATM.REQUEST.CC.CICS) +
    CFSTRUCT(APPLICATION1) +
    QSGDISP(SHARED) +
    DEFSOPTS(SHARED) +

```

```
CLUSTER(CCLUS1) +
SHARE
```

*CHANNELS:

Define one each of these Cluster sender channels on each QMGR*

```
DEFINE CHANNEL(TO.CCLUS1.MQAI2) +
  CHLTYPE(CLUSSDR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQAI2) +
  CHLTYPE(CLUSSDR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(ECLUS1)
```

DEFINE A QMGR CLUSTER RECEIVER CHANNEL
ON THE INDIVIDUAL QMGRS TO CONTAIN THE
QMGRS IPADDR AND PORT. CLUSTER RECEIVER CHANNELS SHOULD
NOT BE SET UP TO USE A SHARED IPADDR OR PORT AS IT CAN CAUSE
ERRORS IN THE REPOSITORY.

Qmgr MQZ1

```
DEFINE CHANNEL(TO.CCLUS1.MQZ1) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  CONNAME('9.12.6.106(1543)') +
  CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQZ1) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  CONNAME('9.12.6.106(1543)') +
  CLUSTER(ECLUS1)
```

Qmgr MQZ2

```
DEFINE CHANNEL(TO.CCLUS1.MQZ2) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
  CONNAME('9.12.6.107(1541)') +
  CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQZ2) +
  CHLTYPE(CLUSRCVR) +
  QSGDISP(QMGR) +
  TRPTYPE(TCP) +
```



```

        CONNAME('9.12.6.107(1541)') +
        CLUSTER(ECLUS1)
Qmgr MQZ3
DEFINE CHANNEL(TO.CCLUS1.MQZ3) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.12.6.108(1542)') +
        CLUSTER(CCLUS1)
DEFINE CHANNEL(TO.ECLUS1.MQZ3) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        CONNAME('9.12.6.108(1542)') +
        CLUSTER(ECLUS1)

```

CLISTS for RACF definitions and access levels for initial setup

The following CLISTS were used to create the initial RACF security setup for the QSG MQZG.

The switch profiles are included at this stage so that security is set to OFF on the Qmgrs within the QSG while the initial setup is completed. Once it is complete, the Switch profiles will be removed.

Example: B-2 MQADMIN class - CLIST to create the initial Switch, reslevel, context, alternate user and command resource profiles

```

00000010CONTROL ASIS
00000020RDEFINE MQADMIN MQZG.NO.SUBSYS.SECURITY UACC(NONE)
00000030RDEFINE MQADMIN MQZG.NO.QMGR.CHECKS UACC(NONE)
00000040RDEFINE MQADMIN MQZG.NO.CONNECT.CHECKS UACC(NONE)
00000050RDEFINE MQADMIN MQZG.NO.CMD.CHECKS UACC(NONE)
00000060RDEFINE MQADMIN MQZG.NO.CMD.RESC.CHECKS UACC(NONE)
00000070RDEFINE MQADMIN MQZG.NO.PROCESS.CHECKS UACC(NONE)
00000080RDEFINE MQADMIN MQZG.NO.QUEUE.CHECKS UACC(NONE)
00000090RDEFINE MQADMIN MQZG.NO.NLIST.CHECKS UACC(NONE)
00000100RDEFINE MQADMIN MQZG.NO.ALTERNATE.USER.CHECKS UACC(NONE)
00000200RDEFINE MQADMIN MQZG.NO.CONTEXT.CHECKS UACC(NONE)
00000210RDEFINE MQADMIN MQZG.RESLEVEL UACC(NONE)
00000300RDEFINE MQADMIN MQZG.QUEUE.* UACC(NONE)
00000301RDEFINE MQADMIN MQZG.PROCESS.* UACC(NONE)
00000302RDEFINE MQADMIN MQZG.NAMELIST.* UACC(NONE)
00000303RDEFINE MQADMIN MQZG.CHANNEL.* UACC(NONE)
00000310RDEFINE MQADMIN MQZG.CONTEXT.* UACC(NONE)
00000320RDEFINE MQADMIN MQZG.ALTERNATE.USER.* UACC(NONE)

```

Example: B-3 MQADMIN class - CLIST to grant initial access of NONE to most users against the profiles previously defined in Appendix B-2, "MQADMIN class - CLIST to create the initial Switch, reslevel, context, alternate user and command resource profiles" on page 269

```

00000010CONTROL ASIS
00000120PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(SDRES1) ACCESS(NONE)
00000121PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000122PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000123PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000124PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000125PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(CICSPA1) ACCESS(NONE)
00000126PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(CICSPA2) ACCESS(NONE)
00000127PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(CICSPA3) ACCESS(NONE)
00000128PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ1MSTR) ACCESS(NONE)
00000129PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ2MSTR) ACCESS(NONE)
00000130PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ3MSTR) ACCESS(NONE)
00000140PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ1CHIN) ACCESS(NONE)
00000150PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ2CHIN) ACCESS(NONE)
00000151PERMIT MQZG.RESLEVEL CLASS(MQADMIN) ID(MQZ3CHIN) ACCESS(NONE)
00000160PERMIT MQZG.CHANNEL.* CLASS(MQADMIN) ID(SDRES1) ACCESS(ALTER)
00000161PERMIT MQZG.CHANNEL.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000162PERMIT MQZG.CHANNEL.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000163PERMIT MQZG.CHANNEL.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000164PERMIT MQZG.CHANNEL.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000170PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(SDRES1) ACCESS(ALTER)
00000171PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000172PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000173PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000174PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000175PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(MQZ1CHIN) ACCESS(ALTER)
00000176PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(MQZ2CHIN) ACCESS(ALTER)
00000177PERMIT MQZG.CONTEXT.* CLASS(MQADMIN) ID(MQZ3CHIN) ACCESS(ALTER)
00000178PERMIT MQZG.ALTERNATE.USER.* CLASS(MQADMIN) ID(SDRES1)
ACCESS(ALTER)
00000179PERMIT MQZG.ALTERNATE.USER.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000180PERMIT MQZG.ALTERNATE.USER.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000181PERMIT MQZG.ALTERNATE.USER.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000182PERMIT MQZG.ALTERNATE.USER.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000183PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ID(SDRES1) ACCESS(ALTER)
00000184PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000185PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000186PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000187PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000190PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ID(SDRES1) ACCESS(ALTER)
00000191PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000192PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000193PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000194PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)

```

```

00000200PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(SDRES1) ACCESS(ALTER)
00000300PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(SDRES2) ACCESS(NONE)
00000400PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(SDRES3) ACCESS(NONE)
00000500PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(SDRES4) ACCESS(NONE)
00000600PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(SDRES5) ACCESS(NONE)
00000700PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(MQZ1CHIN) ACCESS(ALTER)
00000800PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(MQZ2CHIN) ACCESS(ALTER)
00000900PERMIT MQZG.QUEUE.* CLASS(MQADMIN) ID(MQZ3CHIN) ACCESS(ALTER)

```

Example: B-4 MQCMDS class - CLIST to create the initial Command profiles

```

00000010CONTROL ASIS
00000040RDEFINE MQCMDS MQZG.ALTER.* UACC(NONE)
00000050RDEFINE MQCMDS MQZG.ARCHIVE.* UACC(NONE)
00000060RDEFINE MQCMDS MQZG.CLEAR.* UACC(NONE)
00000070RDEFINE MQCMDS MQZG.DEFINE.* UACC(NONE)
00000071RDEFINE MQCMDS MQZG.DELETE.* UACC(NONE)
00000080RDEFINE MQCMDS MQZG.DISPLAY.* UACC(NONE)
00000090RDEFINE MQCMDS MQZG.MOVE.* UACC(NONE)
00000100RDEFINE MQCMDS MQZG.PING.* UACC(NONE)
00000200RDEFINE MQCMDS MQZG.RECOVER.* UACC(NONE)
00000300RDEFINE MQCMDS MQZG.REFRESH.* UACC(NONE)
00000400RDEFINE MQCMDS MQZG.RESET.* UACC(NONE)
00000500RDEFINE MQCMDS MQZG.RESOLVE.* UACC(NONE)
00000600RDEFINE MQCMDS MQZG.RESUME.* UACC(NONE)
00000700RDEFINE MQCMDS MQZG.RVERIFY.* UACC(NONE)
00000800RDEFINE MQCMDS MQZG.SET.* UACC(NONE)
00000900RDEFINE MQCMDS MQZG.START.* UACC(NONE)
00001000RDEFINE MQCMDS MQZG.STOP.* UACC(NONE)
00001100RDEFINE MQCMDS MQZG.SUSPEND.* UACC(NONE)

```

Example: B-5 MQCMDS class - CLIST to grant access of NONE to most users of the profiles previously defined in Appendix B-4, "MQCMDS class - CLIST to create the initial Command profiles" on page 271

```

00000010CONTROL ASIS
00000040PERMIT MQZG.ALTER.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000041PERMIT MQZG.ALTER.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000042PERMIT MQZG.ALTER.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000043PERMIT MQZG.ALTER.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000044PERMIT MQZG.ALTER.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000050PERMIT MQZG.ARCHIVE.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000051PERMIT MQZG.ARCHIVE.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000052PERMIT MQZG.ARCHIVE.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000053PERMIT MQZG.ARCHIVE.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000054PERMIT MQZG.ARCHIVE.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)

```

```

00000055PERMIT MQZG.BACKUP.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000056PERMIT MQZG.BACKUP.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000057PERMIT MQZG.BACKUP.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000058PERMIT MQZG.BACKUP.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000059PERMIT MQZG.BACKUP.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000060PERMIT MQZG.CLEAR.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000061PERMIT MQZG.CLEAR.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000062PERMIT MQZG.CLEAR.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000063PERMIT MQZG.CLEAR.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000064PERMIT MQZG.CLEAR.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000070PERMIT MQZG.DEFINE.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000071PERMIT MQZG.DEFINE.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000072PERMIT MQZG.DEFINE.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000073PERMIT MQZG.DEFINE.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000074PERMIT MQZG.DEFINE.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000080PERMIT MQZG.DELETE.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000081PERMIT MQZG.DELETE.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000082PERMIT MQZG.DELETE.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000083PERMIT MQZG.DELETE.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000084PERMIT MQZG.DELETE.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000090PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(SDRES1) ACCESS(READ)
00000091PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000092PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000093PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000094PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000094PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(MQZ1CHIN) ACCESS(READ)
00000094PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(MQZ2CHIN) ACCESS(READ)
00000094PERMIT MQZG.DISPLAY.* CLASS(MQCMDS) ID(MQZ3CHIN) ACCESS(READ)
00000100PERMIT MQZG.MOVE.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000101PERMIT MQZG.MOVE.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000102PERMIT MQZG.MOVE.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000103PERMIT MQZG.MOVE.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000104PERMIT MQZG.MOVE.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000110PERMIT MQZG.PING.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000111PERMIT MQZG.PING.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000112PERMIT MQZG.PING.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000113PERMIT MQZG.PING.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000114PERMIT MQZG.PING.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000120PERMIT MQZG.RECOVER.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000121PERMIT MQZG.RECOVER.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000122PERMIT MQZG.RECOVER.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000123PERMIT MQZG.RECOVER.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000124PERMIT MQZG.RECOVER.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000130PERMIT MQZG.REFRESH.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000131PERMIT MQZG.REFRESH.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000132PERMIT MQZG.REFRESH.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000133PERMIT MQZG.REFRESH.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000134PERMIT MQZG.REFRESH.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000140PERMIT MQZG.RESET.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)

```

```

00000141PERMIT MQZG.RESET.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000142PERMIT MQZG.RESET.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000143PERMIT MQZG.RESET.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000144PERMIT MQZG.RESET.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000150PERMIT MQZG.RESOLVE.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000151PERMIT MQZG.RESOLVE.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000152PERMIT MQZG.RESOLVE.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000153PERMIT MQZG.RESOLVE.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000154PERMIT MQZG.RESOLVE.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000160PERMIT MQZG.RESUME.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000161PERMIT MQZG.RESUME.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000162PERMIT MQZG.RESUME.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000163PERMIT MQZG.RESUME.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000164PERMIT MQZG.RESUME.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000170PERMIT MQZG.RVERIFY.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000171PERMIT MQZG.RVERIFY.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000172PERMIT MQZG.RVERIFY.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000173PERMIT MQZG.RVERIFY.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000174PERMIT MQZG.RVERIFY.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000180PERMIT MQZG.SET.* CLASS(MQCMDS) ID(SDRES1) ACCESS(ALTER)
00000181PERMIT MQZG.SET.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000182PERMIT MQZG.SET.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000183PERMIT MQZG.SET.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000184PERMIT MQZG.SET.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000190PERMIT MQZG.START.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000190PERMIT MQZG.START.* CLASS(MQCMDS) ID(MQZ1CHIN) ACCESS(CONTROL)
00000190PERMIT MQZG.START.* CLASS(MQCMDS) ID(MQZ2CHIN) ACCESS(CONTROL)
00000190PERMIT MQZG.START.* CLASS(MQCMDS) ID(MQZ3CHIN) ACCESS(CONTROL)
00000191PERMIT MQZG.START.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000192PERMIT MQZG.START.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000193PERMIT MQZG.START.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000194PERMIT MQZG.START.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000200PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000190PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(MQZ1CHIN) ACCESS(CONTROL)
00000190PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(MQZ2CHIN) ACCESS(CONTROL)
00000190PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(MQZ3CHIN) ACCESS(CONTROL)
00000201PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000202PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000203PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000204PERMIT MQZG.STOP.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)
00000210PERMIT MQZG.SUSPEND.* CLASS(MQCMDS) ID(SDRES1) ACCESS(CONTROL)
00000220PERMIT MQZG.SUSPEND.* CLASS(MQCMDS) ID(SDRES2) ACCESS(NONE)
00000230PERMIT MQZG.SUSPEND.* CLASS(MQCMDS) ID(SDRES3) ACCESS(NONE)
00000240PERMIT MQZG.SUSPEND.* CLASS(MQCMDS) ID(SDRES4) ACCESS(NONE)
00000250PERMIT MQZG.SUSPEND.* CLASS(MQCMDS) ID(SDRES5) ACCESS(NONE)

```

Example: B-6 MQCONN Class - Clist to create connection profiles

```
00000010CONTROL ASIS
00000020DEFINE MQCONN MQZG.BATCH UACC(NONE)
00000030DEFINE MQCONN MQZG.CICS UACC(NONE)
00000040DEFINE MQCONN MQZG.IMS UACC(NONE)
00000050DEFINE MQCONN MQZG.CHIN UACC(NONE)
```

Example: B-7 MQCONN Class - CLIST to grant initial access of NONE to most users to the profiles previously defined in Appendix B-6, "MQCONN Class - Clist to create connection profiles" on page 274

```
00000010CONTROL ASIS
00000020PERMIT MQZG.BATCH CLASS(MQCONN) ID(SDRES1) ACCESS(NONE)
00000021PERMIT MQZG.BATCH CLASS(MQCONN) ID(SDRES2) ACCESS(NONE)
00000022PERMIT MQZG.BATCH CLASS(MQCONN) ID(SDRES3) ACCESS(NONE)
00000023PERMIT MQZG.BATCH CLASS(MQCONN) ID(SDRES4) ACCESS(NONE)
00000024PERMIT MQZG.BATCH CLASS(MQCONN) ID(SDRES5) ACCESS(NONE)
00000030PERMIT MQZG.CHIN CLASS(MQCONN) ID(MQZ1CHIN) ACCESS(READ)
00000031PERMIT MQZG.CHIN CLASS(MQCONN) ID(MQZ2CHIN) ACCESS(READ)
00000032PERMIT MQZG.CHIN CLASS(MQCONN) ID(MQZ3CHIN) ACCESS(READ)
00000040PERMIT MQZG.CICS CLASS(MQCONN) ID(CICSPA1) ACCESS(READ)
00000041PERMIT MQZG.CICS CLASS(MQCONN) ID(CICSPA2) ACCESS(READ)
00000042PERMIT MQZG.CICS CLASS(MQCONN) ID(CICSPA3) ACCESS(READ)
```

Example: B-8 MQQUEUE Class - CLIST to create initial profiles for Queues

```
00000010CONTROL ASIS
00000020DEFINE MQQUEUE MQZG.SYSTEM.* UACC(NONE)
00000030DEFINE MQQUEUE MQZG.CSQ4SAMP.* UACC(NONE)
00000040DEFINE MQQUEUE MQZG.** UACC(NONE)
```

Example: B-9 MQQUEUE Class - CLIST to grant initial access of NONE to most users to the profiles previously defined in Example B-8

```
00000010CONTROL ASIS
00000020PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ID(SDRES1) ACCESS(ALTER)
00000021PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ID(SDRES2) ACCESS(NONE)
00000022PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ID(SDRES3) ACCESS(NONE)
00000023PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ID(SDRES4) ACCESS(NONE)
00000024PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ID(SDRES5) ACCESS(NONE)
00000030PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES) ACCESS(ALTER)
00000030PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES1) ACCESS(ALTER)
00000031PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES2) ACCESS(NONE)
00000032PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES3) ACCESS(NONE)
00000033PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES4) ACCESS(NONE)
00000034PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(SDRES4) ACCESS(NONE)
```

```

00000035PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ1MSTR) ACCESS(NONE)
00000036PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ2MSTR) ACCESS(NONE)
00000037PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ3MSTR) ACCESS(NONE)
00000038PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ1CHIN) ACCESS(ALTER)
00000039PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ2CHIN) ACCESS(ALTER)
00000040PERMIT MQZG.SYSTEM.** CLASS(MQQUEUE) ID(MQZ3CHIN) ACCESS(ALTER)
00000041PERMIT MQZG.** CLASS(MQQUEUE) ID(SDRES1) ACCESS(ALTER)
00000050PERMIT MQZG.** CLASS(MQQUEUE) ID(SDRES2) ACCESS(NONE)
00000060PERMIT MQZG.** CLASS(MQQUEUE) ID(SDRES3) ACCESS(NONE)
00000070PERMIT MQZG.** CLASS(MQQUEUE) ID(SDRES4) ACCESS(NONE)
00000080PERMIT MQZG.** CLASS(MQQUEUE) ID(SDRES5) ACCESS(NONE)
00000090PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ1MSTR) ACCESS(ALTER)
00000100PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ2MSTR) ACCESS(ALTER)
00000200PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ3MSTR) ACCESS(ALTER)
00000300PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ1CHIN) ACCESS(ALTER)
00000400PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ2CHIN) ACCESS(ALTER)
00000500PERMIT MQZG.** CLASS(MQQUEUE) ID(MQZ3CHIN) ACCESS(ALTER)
00000600PERMIT MQZG.** CLASS(MQQUEUE) ID(CICSPA Z1) ACCESS(NONE)
00000700PERMIT MQZG.** CLASS(MQQUEUE) ID(CICSPA Z2) ACCESS(NONE)
00000800PERMIT MQZG.** CLASS(MQQUEUE) ID(CICSPA Z3) ACCESS(NONE)

```

Example: B-10 MQPROC Class - Clist to define initial profile for processes

```

00000010CONTROL ASIS
00000020DEFINE MQPROC MQZG.** UACC(NONE)

```

Example: B-11 MQPROC Class - CLIST to grant initial acces of NONE to all users to the profile previously defined in Example B-10

```

00000010CONTROL ASIS
00000020PERMIT MQZG.** CLASS(MQPROC) ID(SDRES1) ACCESS(NONE)
00000030PERMIT MQZG.** CLASS(MQPROC) ID(SDRES2) ACCESS(NONE)
00000040PERMIT MQZG.** CLASS(MQPROC) ID(SDRES3) ACCESS(NONE)
00000050PERMIT MQZG.** CLASS(MQPROC) ID(SDRES4) ACCESS(NONE)
00000060PERMIT MQZG.** CLASS(MQPROC) ID(SDRES5) ACCESS(NONE)

```

Example: B-12 MQNLIST Class = CLIST to define profile for namelists

```

00000010CONTROL ASIS
00000020DEFINE MQPROC MQZG.** UACC(NONE)

```

Example: B-13 MQNLIST Class - CLIST to grant initial access of NONE to all users for the profile previously defined in Appendix B-12, "MQNLIST Class = CLIST to define profile for namelists" on page 275

```
00000010CONTROL ASIS
00000020PERMIT MQZG.** CLASS(MQPROC) ID(SDRES1) ACCESS(NONE)
00000030PERMIT MQZG.** CLASS(MQPROC) ID(SDRES2) ACCESS(NONE)
00000040PERMIT MQZG.** CLASS(MQPROC) ID(SDRES3) ACCESS(NONE)
00000050PERMIT MQZG.** CLASS(MQPROC) ID(SDRES4) ACCESS(NONE)
00000060PERMIT MQZG.** CLASS(MQPROC) ID(SDRES5) ACCESS(NONE)
```

CLISTS for RACF definitions and access levels for business scenario

The following CLISTS show the definitions and access levels for the RACF profiles required for the Business scenario solution. The access levels granted to different user IDs have been tailored to allow access to only those user IDs that need it.

Example: B-14 MQADMIN class - Clist showing initial RACF definitions for the business scenario solution

```
00000010CONTROL ASIS
00000020DELETE MQADMIN MQZG.NO.SUBSYS.SECURITY
00000030DELETE MQADMIN MQZG.NO.QSG.CHECKS
00000040DELETE MQADMIN MQZG.NO.CONNECT.CHECKS
00000050DELETE MQADMIN MQZG.NO.CMD.CHECKS
00000060DELETE MQADMIN MQZG.NO.CMD.RESC.CHECKS
00000070DELETE MQADMIN MQZG.NO.PROCESS.CHECKS
00000080DELETE MQADMIN MQZG.NO.QUEUE.CHECKS
00000090DELETE MQADMIN MQZG.NO.NLIST.CHECKS
00000100DELETE MQADMIN MQZG.NO.ALTERNATE.USER.CHECKS
00000200DELETE MQADMIN MQZG.NO.CONTEXT.CHECKS
00000300DELETE MQADMIN MQZG.CONTEXT.*
00000400DELETE MQADMIN MQZG.ALTERNATE.USER.*
00000500DELETE MQADMIN MQZG.CHANNEL.*
00000600DELETE MQADMIN MQZG.QUEUE.*
00000700DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES UACC(NONE)
00000800DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES1 UACC(NONE)
00000900DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES2 UACC(NONE)
00001000DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES3 UACC(NONE)
00001100DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES4 UACC(NONE)
00001200DEFINE MQADMIN MQZG.ALTERNATE.USER.SDRES5 UACC(NONE)
00001210DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ1MSTR UACC(NONE)
00001220DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ2MSTR UACC(NONE)
00001230DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ3MSTR UACC(NONE)
00001240DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ1CHIN UACC(NONE)
00001250DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ2CHIN UACC(NONE)
00001260DEFINE MQADMIN MQZG.ALTERNATE.USER.MQZ3CHIN UACC(NONE)
```


00001300RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT UACC(NONE)
 00001310RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT UACC(NONE)
 00001320RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT UACC(NONE)
 00001330RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.ADMIN.QMGR.EVENT UACC(NONE)
 00001340RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ UACC(NONE)
 00001350RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ UACC(NONE)
 00001360RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CLUSTER.COMMAND.QUEUE
 UACC(NONE)
 00001370RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CLUSTER.REPOSITORY.QUEUE
 UACC(NONE)
 00001380RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.QUEUE
 UACC(NONE)
 00001390RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.COMMAND.INPUT UACC(NONE)
 00001391RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.COMMAND.REPLY.MODEL UACC(NONE)
 00001392RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.QSG.CHANNEL.SYNCQ UACC(NONE)
 00001393RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.QSG.TRANSMIT.QUEUE UACC(NONE)
 00001394RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CSQOREXX.* UACC(NONE)
 00001395RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CSQUTIL.* UACC(NONE)
 00001396RDEFINE MQADMIN MQZG.CONTEXT.SYSTEM.CSQXCMD.* UACC(NONE)
 00001400RDEFINE MQADMIN MQZG.CONTEXT.CSQ4SAMP.* UACC(NONE)
 00001410RDEFINE MQADMIN MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS UACC(NONE)
 00001420RDEFINE MQADMIN MQZG.CONTEXT.ATM.REQUEST.CC.CICS UACC(NONE)
 00001430RDEFINE MQADMIN MQZG.CONTEXT.CICS01.INITQ UACC(NONE)
 00001440RDEFINE MQADMIN MQZG.CONTEXT.DLQ.MQZ1 UACC(NONE)
 00001440RDEFINE MQADMIN MQZG.CONTEXT.DLQ.MQZ2 UACC(NONE)
 00001440RDEFINE MQADMIN MQZG.CONTEXT.DLQ.MQZ3 UACC(NONE)
 00001450RDEFINE MQADMIN MQZG.CONTEXT.MQ400 UACC(NONE)
 00001460RDEFINE MQADMIN MQZG.CONTEXT.MQ400.SC UACC(NONE)
 00001470RDEFINE MQADMIN MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED
 UACC(NONE)
 00001480RDEFINE MQADMIN MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED.SC
 UACC(NONE)
 00001490RDEFINE MQADMIN MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 UACC(NONE)
 00001491RDEFINE MQADMIN MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC UACC(NONE)
 00001492RDEFINE MQADMIN MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS UACC(NONE)
 00001500RDEFINE MQADMIN MQZG.QUEUE.ACCOUNT.REQUEST.EC.CICS UACC(NONE)
 00001510RDEFINE MQADMIN MQZG.QUEUE.ATM.REQUEST.CC.CICS UACC(NONE)
 00001520RDEFINE MQADMIN MQZG.QUEUE.CICS01.INITQ UACC(NONE)
 00001530RDEFINE MQADMIN MQZG.QUEUE.DLQ.MQZ1 UACC(NONE)
 00001530RDEFINE MQADMIN MQZG.QUEUE.DLQ.MQZ2 UACC(NONE)
 00001530RDEFINE MQADMIN MQZG.QUEUE.DLQ.MQZ3 UACC(NONE)
 00001540RDEFINE MQADMIN MQZG.QUEUE.MQ400 UACC(NONE)
 00001550RDEFINE MQADMIN MQZG.QUEUE.MQ400.SC UACC(NONE)
 00001560RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.BATCH.COMPLETE.SHARED
 UACC(NONE)
 00001570RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.BATCH.COMPLETE.SHARED.SC
 UACC(NONE)
 00001580RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.BATCH.MQ400 UACC(NONE)
 00001590RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.BATCH.MQ400.SC UACC(NONE)

```

00001592RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.REQUEST.EC.CICS UACC(NONE)
00001593RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.ADMIN.CHANNEL.EVENT UACC(NONE)
00001594RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.ADMIN.CONFIG.EVENT UACC(NONE)
00001595RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.ADMIN.PERFM.EVENT UACC(NONE)
00001596RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.ADMIN.QMGR.EVENT UACC(NONE)
00001597RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CHANNEL.INITQ UACC(NONE)
00001598RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CHANNEL.SYNCQ UACC(NONE)
00001599RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CLUSTER.COMMAND.QUEUE UACC(NONE)
00001600RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CLUSTER.REPOSITORY.QUEUE
UACC(NONE)
00001601RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CLUSTER.TRANSMIT.QUEUE UACC(NONE)
00001602RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.COMMAND.INPUT UACC(NONE)
00001603RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.COMMAND.REPLY.MODEL UACC(NONE)
00001604RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.QSG.CHANNEL.SYNCQ UACC(NONE)
00001605RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.QSG.TRANSMIT.QUEUE UACC(NONE)
00001606RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CSQOREXX.* UACC(NONE)
00001608RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CSQUTIL.* UACC(NONE)
00001609RDEFINE MQADMIN MQZG.QUEUE.SYSTEM.CSQXCMD.* UACC(NONE)
00001610RDEFINE MQADMIN MQZG.QUEUE.CSQ4SAMP.* UACC(NONE)
00001611RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.REQUEST.EC.CICS UACC(NONE)
00001612RDEFINE MQADMIN MQZG.QUEUE.PORTFOLIO.REQUEST.EC.CICS UACC(NONE)
00001613RDEFINE MQADMIN MQZG.CHANNEL.MQW2K3.CC.MQZG UACC(NONE)
00001620RDEFINE MQADMIN MQZG.CHANNEL.MQZG.CC.MQ400 UACC(NONE)
00001630RDEFINE MQADMIN MQZG.CHANNEL.MQZG.SC.MQ400 UACC(NONE)
00001640RDEFINE MQADMIN MQZG.CHANNEL.MQ400.CC.MQZG UACC(NONE)
00001650RDEFINE MQADMIN MQZG.CHANNEL.MQ400.SC.MQZG UACC(NONE)
00001660RDEFINE MQADMIN MQZG.CHANNEL.TO.CCLUS1.MQAIX2 UACC(NONE)
00001670RDEFINE MQADMIN MQZG.CHANNEL.TO.ECLUS1.MQAIX2 UACC(NONE)
00001680RDEFINE MQADMIN MQZG.CHANNEL.TO.CCLUS1.MQZ1 UACC(NONE)
00001690RDEFINE MQADMIN MQZG.CHANNEL.TO.ECLUS1.MQZ1 UACC(NONE)
00001680RDEFINE MQADMIN MQZG.CHANNEL.TO.CCLUS1.MQZ2 UACC(NONE)
00001690RDEFINE MQADMIN MQZG.CHANNEL.TO.ECLUS1.MQZ2 UACC(NONE)
00001680RDEFINE MQADMIN MQZG.CHANNEL.TO.CCLUS1.MQZ3 UACC(NONE)
00001690RDEFINE MQADMIN MQZG.CHANNEL.TO.ECLUS1.MQZ3 UACC(NONE)
00001900RDEFINE MQADMIN MQZG.PROCESS.* UACC(NONE)
00002000RDEFINE MQADMIN MQZG.NAMELIST.* UACC(NONE)

```

Example: B-15 MQADMIN Class - Clist showing the initial access levels granted against the profiles previously defined in Appendix B-14, "MQADMIN class - Clist showing initial RACF definitions for the business scenario solution" on page 276

```

00000010CONTROL ASIS
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)

```

```

00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001300PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001310PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001320PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001330PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001330PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001330PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001330PERMIT MQZG.CONTEXT.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)

```

```

00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001340PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001350PERMIT MQZG.CONTEXT.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001360PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001360PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001360PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001360PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
00001370PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001370PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
00001380PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001380PERMIT MQZG.CONTEXT.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
00001390PERMIT MQZG.CONTEXT.SYSTEM.COMMAND.INPUT CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001391PERMIT MQZG.CONTEXT.SYSTEM.COMMAND.INPUT CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001393PERMIT MQZG.CONTEXT.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001394PERMIT MQZG.CONTEXT.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)

```

```

00001396PERMIT MQZG.CONTEXT.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001396PERMIT MQZG.CONTEXT.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
00001397PERMIT MQZG.CONTEXT.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001397PERMIT MQZG.CONTEXT.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
00001398PERMIT MQZG.CONTEXT.SYSTEM.CSQOREXX.* CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES1)
00001399PERMIT MQZG.CONTEXT.SYSTEM.CSQUTIL.* CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES1)
00001400PERMIT MQZG.CONTEXT.SYSTEM.CSQXCMD.* CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES1)
00001400PERMIT MQZG.CONTEXT.SYSTEM.CSQXCMD.* CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001400PERMIT MQZG.CONTEXT.SYSTEM.CSQXCMD.* CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001400PERMIT MQZG.CONTEXT.SYSTEM.CSQXCMD.* CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001401PERMIT MQZG.CONTEXT.CSQ4SAMP.* CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES1)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES1)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES2)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES4)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(SDRES5)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ1CHIN)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001410PERMIT MQZG.CONTEXT.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES1)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES2)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES4)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(SDRES5)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)

```

```

00001420PERMIT MQZG.CONTEXT.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001430PERMIT MQZG.CONTEXT.CICS01.INITQ CLASS(MQADMIN) ACC(NONE)
ID(SDRES1)
00001440PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1)
00001441PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES2)
00001442PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES4)
00001443PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES5)
00001444PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001445PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001446PERMIT MQZG.CONTEXT.DLQ.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001447PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1)
00001448PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES2)
00001449PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES4)
00001450PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES5)
00001451PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001452PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001453PERMIT MQZG.CONTEXT.DLQ.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001454PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1)
00001455PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES2)
00001456PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES4)
00001457PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES5)
00001458PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001459PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001460PERMIT MQZG.CONTEXT.DLQ.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001461PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1)
00001462PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES2)
00001463PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES4)
00001464PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES5)
00001465PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(MQZ1CHIN)
00001466PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(MQZ2CHIN)
00001467PERMIT MQZG.CONTEXT.MQ400 CLASS(MQADMIN) ACC(CONTROL) ID(MQZ3CHIN)
00001468PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1)
00001469PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL) ID(SDRES2)
00001470PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL) ID(SDRES4)
00001471PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL) ID(SDRES5)
00001472PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001473PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)

```

00001474PERMIT MQZG.CONTEXT.MQ400.SC CLASS(MQADMIN) ACC(CONTROL)
 ID(MQZ3CHIN)
 00001470PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
 00001470PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
 00001480PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED.SC
 CLASS(MQADMIN) ACC(CONTROL) ID(SDRES1,SDRES2,SDRES4,SDRES5)
 00001480PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.COMPLETE.SHARED.SC
 CLASS(MQADMIN) ACC(CONTROL) ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES1)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES2)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES4)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES5)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ1CHIN)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ2CHIN)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ3CHIN)
 00001491PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES1)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES2)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES4)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES5)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ1CHIN)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ2CHIN)
 00001490PERMIT MQZG.CONTEXT.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ3CHIN)
 00001492PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES1)
 00001493PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES2)
 00001494PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES4)
 00001494PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
 ACC(CONTROL) ID(SDRES5)
 00001494PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
 ACC(CONTROL) ID(MQZ1CHIN)

```

00001494PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ2CHIN)
00001494PERMIT MQZG.CONTEXT.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(CONTROL) ID(MQZ3CHIN)
00001500PERMIT MQZG.QUEUE.ACCOUNT.REQUEST.EC.CICS CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001510PERMIT MQZG.QUEUE.ATM.REQUEST.CC.CICS CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001520PERMIT MQZG.QUEUE.CICS01.INITQ CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00001530PERMIT MQZG.QUEUE.DLQ.MQZ1 CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00001530PERMIT MQZG.QUEUE.DLQ.MQZ2 CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00001530PERMIT MQZG.QUEUE.DLQ.MQZ3 CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)

00001540PERMIT MQZG.QUEUE.MQ400 CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00001550PERMIT MQZG.QUEUE.MQ400.SC CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00001560PERMIT MQZG.QUEUE.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001570PERMIT MQZG.QUEUE.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001580PERMIT MQZG.QUEUE.PORTFOLIO.BATCH.MQ400 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001590PERMIT MQZG.QUEUE.PORTFOLIO.BATCH.MQ400.SC CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001592PERMIT MQZG.QUEUE.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001593PERMIT MQZG.QUEUE.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001594PERMIT MQZG.QUEUE.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001595PERMIT MQZG.QUEUE.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001596PERMIT MQZG.QUEUE.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001597PERMIT MQZG.QUEUE.SYSTEM.CHANNEL.INITQ CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001598PERMIT MQZG.QUEUE.SYSTEM.CHANNEL.SYNCQ CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001599PERMIT MQZG.QUEUE.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001600PERMIT MQZG.QUEUE.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQADMIN)
ACC(UPDATE) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001601PERMIT MQZG.QUEUE.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001602PERMIT MQZG.QUEUE.SYSTEM.COMMAND.INPUT CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1)
00001603PERMIT MQZG.QUEUE.SYSTEM.COMMAND.INPUT CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)

```



```

00001604PERMIT MQZG.QUEUE.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQADMIN)
ACC(UPDATE) ID(SDRES1)
00001605PERMIT MQZG.QUEUE.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQADMIN)
ACC(UPDATE) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001606PERMIT MQZG.QUEUE.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQADMIN)
ACC(UPDATE) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001607PERMIT MQZG.QUEUE.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQADMIN)
ACC(UPDATE) ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001608PERMIT MQZG.QUEUE.SYSTEM.CSQOREXX.* CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1)
00001609PERMIT MQZG.QUEUE.SYSTEM.CSQUTIL.* CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1)
00001610PERMIT MQZG.QUEUE.SYSTEM.CSQXCMD.* CLASS(MQADMIN) ACC(UPDATE)
ID(SDRES1,MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00001611PERMIT MQZG.QUEUE.CSQ4SAMP.* CLASS(MQADMIN) ACC(UPDATE)ID(SDRES1)
00001612PERMIT MQZG.QUEUE.PORTFOLIO.REQUEST.EC.CICS CLASS(MQADMIN)
ACC(ALTER) ID(SDRES1)
00001614PERMIT MQZG.CHANNEL.MQW2K3.CC.MQZG CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001614PERMIT MQZG.CHANNEL.MQW2K3.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001614PERMIT MQZG.CHANNEL.MQW2K3.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001614PERMIT MQZG.CHANNEL.MQW2K3.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001620PERMIT MQZG.CHANNEL.MQZG.CC.MQ400 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001620PERMIT MQZG.CHANNEL.MQZG.CC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001620PERMIT MQZG.CHANNEL.MQZG.CC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001620PERMIT MQZG.CHANNEL.MQZG.CC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001630PERMIT MQZG.CHANNEL.MQZG.SC.MQ400 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001630PERMIT MQZG.CHANNEL.MQZG.SC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001630PERMIT MQZG.CHANNEL.MQZG.SC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001630PERMIT MQZG.CHANNEL.MQZG.SC.MQ400 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001640PERMIT MQZG.CHANNEL.MQ400.CC.MQZG CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001640PERMIT MQZG.CHANNEL.MQ400.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001640PERMIT MQZG.CHANNEL.MQ400.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001640PERMIT MQZG.CHANNEL.MQ400.CC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)

```

```

00001650PERMIT MQZG.CHANNEL.MQ400.SC.MQZG CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001650PERMIT MQZG.CHANNEL.MQ400.SC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001650PERMIT MQZG.CHANNEL.MQ400.SC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001650PERMIT MQZG.CHANNEL.MQ400.SC.MQZG CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001660PERMIT MQZG.CHANNEL.TO.CCLUS1.MQAI2 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001660PERMIT MQZG.CHANNEL.TO.CCLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001660PERMIT MQZG.CHANNEL.TO.CCLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001660PERMIT MQZG.CHANNEL.TO.CCLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001670PERMIT MQZG.CHANNEL.TO.ECLUS1.MQAI2 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001660PERMIT MQZG.CHANNEL.TO.ECLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001660PERMIT MQZG.CHANNEL.TO.ECLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001660PERMIT MQZG.CHANNEL.TO.ECLUS1.MQAI2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ1 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ1 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ1 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ2 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)

```

```

00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ2 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ2 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ3 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001680PERMIT MQZG.CHANNEL.TO.CCLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ3 CLASS(MQADMIN) ACC(ALTER)
ID(SDRES1)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ1CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ2CHIN)
00001690PERMIT MQZG.CHANNEL.TO.ECLUS1.MQZ3 CLASS(MQADMIN) ACC(CONTROL)
ID(MQZ3CHIN)
00001900PERMIT MQZG.PROCESS.* CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)
00002000PERMIT MQZG.NAMELIST.* CLASS(MQADMIN) ACC(ALTER) ID(SDRES1)

```

Example: B-16 MQCMDS class - Clist showing initial RACF definitions for the business scenario solution

```

00000010CONTROL ASIS
00000020RDELETE MQCMDS MQZG.ALTER.* UACC(NONE)
00000030RDELETE MQCMDS MQZG.ARCHIVE.* UACC(NONE)
00000031RDELETE MQCMDS MQZG.BACKUP.* UACC(NONE)
00000032RDELETE MQCMDS MQZG.CLEAR.* UACC(NONE)
00000033RDELETE MQCMDS MQZG.DEFINE.* UACC(NONE)
00000034RDELETE MQCMDS MQZG.DELETE.* UACC(NONE)
00000035RDELETE MQCMDS MQZG.DISPLAY.* UACC(NONE)
00000036RDELETE MQCMDS MQZG.MOVE.* UACC(NONE)
00000037RDELETE MQCMDS MQZG.PING.* UACC(NONE)
00000038RDELETE MQCMDS MQZG.RECOVER.* UACC(NONE)
00000039RDELETE MQCMDS MQZG.REFRESH.* UACC(NONE)
00000040RDELETE MQCMDS MQZG.RESET.* UACC(NONE)
00000041RDELETE MQCMDS MQZG.RESOLVE.* UACC(NONE)
00000042RDELETE MQCMDS MQZG.RESUME.* UACC(NONE)
00000043RDELETE MQCMDS MQZG.RVERIFY.* UACC(NONE)
00000044RDELETE MQCMDS MQZG.SET.* UACC(NONE)
00000045RDELETE MQCMDS MQZG.START.* UACC(NONE)
00000046RDELETE MQCMDS MQZG.STOP.* UACC(NONE)

```

00000047RDELETE MQCMDS MQZG.SUSPEND.* UACC(NONE)
 00000049RDEFINE MQCMDS MQZG.ALTER.AUTHINFO UACC(NONE)
 00000050RDEFINE MQCMDS MQZG.ALTER.CFSTRUCT UACC(NONE)
 00000051RDEFINE MQCMDS MQZG.ALTER.CHANNEL UACC(NONE)
 00000052RDEFINE MQCMDS MQZG.ALTER.NAMELIST UACC(NONE)
 00000053RDEFINE MQCMDS MQZG.ALTER.PROCESS UACC(NONE)
 00000054RDEFINE MQCMDS MQZG.ALTER.QALIAS UACC(NONE)
 00000055RDEFINE MQCMDS MQZG.ALTER.QLOCAL UACC(NONE)
 00000056RDEFINE MQCMDS MQZG.ALTER.QMGR UACC(NONE)
 00000057RDEFINE MQCMDS MQZG.ALTER.QMODAL UACC(NONE)
 00000058RDEFINE MQCMDS MQZG.ALTER.QREMOTE UACC(NONE)
 00000059RDEFINE MQCMDS MQZG.ALTER.SECURITY UACC(NONE)
 00000060RDEFINE MQCMDS MQZG.ALTER.STGCLASS UACC(NONE)
 00000061RDEFINE MQCMDS MQZG.ALTER.TRACE UACC(NONE)
 00000062RDEFINE MQCMDS MQZG.ARCHIVE.LOG UACC(NONE)
 00000063RDEFINE MQCMDS MQZG.BACKUP.CFSTRUCT UACC(NONE)
 00000064RDEFINE MQCMDS MQZG.CLEAR.QLOCAL UACC(NONE)
 00000070RDEFINE MQCMDS MQZG.DEFINE.AUTHINFO UACC(NONE)
 00000071RDEFINE MQCMDS MQZG.DEFINE.BUFFPOOL UACC(NONE)
 00000072RDEFINE MQCMDS MQZG.DEFINE.CFSTRUCT UACC(NONE)
 00000073RDEFINE MQCMDS MQZG.DEFINE.CHANNEL UACC(NONE)
 00000074RDEFINE MQCMDS MQZG.DEFINE.MAXSMSGS UACC(NONE)
 00000075RDEFINE MQCMDS MQZG.DEFINE.NAMELIST UACC(NONE)
 00000076RDEFINE MQCMDS MQZG.DEFINE.PROCESS UACC(NONE)
 00000077RDEFINE MQCMDS MQZG.DEFINE.PSID UACC(NONE)
 00000078RDEFINE MQCMDS MQZG.DEFINE.QALIAS UACC(NONE)
 00000079RDEFINE MQCMDS MQZG.DEFINE.QLOCAL UACC(NONE)
 00000080RDEFINE MQCMDS MQZG.DEFINE.QMODEL UACC(NONE)
 00000081RDEFINE MQCMDS MQZG.DEFINE.REMOTE UACC(NONE)
 00000082RDEFINE MQCMDS MQZG.DEFINE.STGCLASS UACC(NONE)
 00000083RDEFINE MQCMDS MQZG.DELETE.AUTHINFO UACC(NONE)
 00000084RDEFINE MQCMDS MQZG.DELETE.CFSTRUCT UACC(NONE)
 00000085RDEFINE MQCMDS MQZG.DELETE.CHANNEL UACC(NONE)
 00000086RDEFINE MQCMDS MQZG.DELETE.NAMELIST UACC(NONE)
 00000087RDEFINE MQCMDS MQZG.DELETE.PROCESS UACC(NONE)
 00000088RDEFINE MQCMDS MQZG.DELETE.QALIAS UACC(NONE)
 00000089RDEFINE MQCMDS MQZG.DELETE.QLOCAL UACC(NONE)
 00000090RDEFINE MQCMDS MQZG.DELETE.QMODEL UACC(NONE)
 00000091RDEFINE MQCMDS MQZG.DELETE.QREMOTE UACC(NONE)
 00000092RDEFINE MQCMDS MQZG.DELETE.STGCLASS UACC(NONE)
 00000093RDEFINE MQCMDS MQZG.DISPLAY.ARCHIVE UACC(NONE)
 00000094RDEFINE MQCMDS MQZG.DISPLAY.AUTHINFO UACC(NONE)
 00000095RDEFINE MQCMDS MQZG.DISPLAY.CFSTATUS UACC(NONE)
 00000096RDEFINE MQCMDS MQZG.DISPLAY.CFSTRUCT UACC(NONE)
 00000097RDEFINE MQCMDS MQZG.DISPLAY.CHANNEL UACC(NONE)
 00000098RDEFINE MQCMDS MQZG.DISPLAY.CHSTATUS UACC(NONE)
 00000099RDEFINE MQCMDS MQZG.DISPLAY.CLUSQMGR UACC(NONE)
 00000100RDEFINE MQCMDS MQZG.DISPLAY.CMDSERV UACC(NONE)
 00000101RDEFINE MQCMDS MQZG.DISPLAY.DQM UACC(NONE)

```

00000102RDEFINE MQCMDS MQZG.DISPLAY.GROUP UACC(NONE)
00000103RDEFINE MQCMDS MQZG.DISPLAY.LOG UACC(NONE)
00000104RDEFINE MQCMDS MQZG.DISPLAY.MAXSMSGS UACC(NONE)
00000105RDEFINE MQCMDS MQZG.DISPLAY.NAMELIST UACC(NONE)
00000106RDEFINE MQCMDS MQZG.DISPLAY.PROCESS UACC(NONE)
00000107RDEFINE MQCMDS MQZG.DISPLAY.QALIAS UACC(NONE)
00000108RDEFINE MQCMDS MQZG.DISPLAY.QCLUSTER UACC(NONE)
00000109RDEFINE MQCMDS MQZG.DISPLAY.QLOCAL UACC(NONE)
00000110RDEFINE MQCMDS MQZG.DISPLAY.QMGR UACC(NONE)
00000111RDEFINE MQCMDS MQZG.DISPLAY.QMODEL UACC(NONE)
00000112RDEFINE MQCMDS MQZG.DISPLAY.QREMOTE UACC(NONE)
00000113RDEFINE MQCMDS MQZG.DISPLAY.QSTATUS UACC(NONE)
00000114RDEFINE MQCMDS MQZG.DISPLAY.QUEUE UACC(NONE)
00000115RDEFINE MQCMDS MQZG.DISPLAY.THREAD UACC(NONE)
00000116RDEFINE MQCMDS MQZG.DISPLAY.SECURITY UACC(NONE)
00000117RDEFINE MQCMDS MQZG.DISPLAY.STGCLASS UACC(NONE)
00000118RDEFINE MQCMDS MQZG.DISPLAY.SYSTEM UACC(NONE)
00000119RDEFINE MQCMDS MQZG.DISPLAY.TRACE UACC(NONE)
00000120RDEFINE MQCMDS MQZG.DISPLAY.USAGE UACC(NONE)
00000121RDEFINE MQCMDS MQZG.MOVE.QLOCAL UACC(NONE)
00000130RDEFINE MQCMDS MQZG.PING.CHANNEL UACC(NONE)
00000200RDEFINE MQCMDS MQZG.RECOVER.BSDS UACC(NONE)
00000210RDEFINE MQCMDS MQZG.RECOVER.CFSTRUCT UACC(NONE)
00000300RDEFINE MQCMDS MQZG.REFRESH.CLUSTER UACC(NONE)
00000310RDEFINE MQCMDS MQZG.REFRESH.QMGR UACC(NONE)
00000320RDEFINE MQCMDS MQZG.REFRESH.SECURITY UACC(NONE)
00000400RDEFINE MQCMDS MQZG.RESET.CHANNEL UACC(NONE)
00000410RDEFINE MQCMDS MQZG.RESET.CLUSTER UACC(NONE)
00000420RDEFINE MQCMDS MQZG.RESET.QSTATS UACC(NONE)
00000430RDEFINE MQCMDS MQZG.RESET.TPIPE UACC(NONE)
00000500RDEFINE MQCMDS MQZG.RESOLVE.CHANNEL UACC(NONE)
00000600RDEFINE MQCMDS MQZG.RESUME.INDOUBT UACC(NONE)
00000700RDEFINE MQCMDS MQZG.RVERIFY.SECURITY UACC(NONE)
00000800RDEFINE MQCMDS MQZG.SET.ARCHIVE UACC(NONE)
00000810RDEFINE MQCMDS MQZG.SET.LOG UACC(NONE)
00000820RDEFINE MQCMDS MQZG.SET.SYSTEM UACC(NONE)
00000900RDEFINE MQCMDS MQZG.START.CHANNEL UACC(NONE)
00000910RDEFINE MQCMDS MQZG.START.CHINIT UACC(NONE)
00000920RDEFINE MQCMDS MQZG.START.CMDSERV UACC(NONE)
00000930RDEFINE MQCMDS MQZG.START.LISTENER UACC(NONE)
00000940RDEFINE MQCMDS MQZG.START.TRACE UACC(NONE)
00001000RDEFINE MQCMDS MQZG.STOP.CHANNEL UACC(NONE)
00001010RDEFINE MQCMDS MQZG.STOP.CHINIT UACC(NONE)
00001020RDEFINE MQCMDS MQZG.STOP.CMDSERV UACC(NONE)
00001030RDEFINE MQCMDS MQZG.STOP.LISTENER UACC(NONE)
00001040RDEFINE MQCMDS MQZG.STOP.QMGR UACC(NONE)
00001050RDEFINE MQCMDS MQZG.STOP.TRACE UACC(NONE)
00001100RDEFINE MQCMDS MQZG.SUSPEND.QMGR UACC(NONE)

```

Example: B-17 MQCMDS class - CLIST to grant initial access to a single administration user ID to the profiles previously defined in Appendix B-16, "MQCMDS class - Clist showing initial RACF definitions for the business scenario solution" on page 287

```

00000010CONTROL ASIS
00000040PERMIT MQZG.ALTER.AUTHINFO CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000041PERMIT MQZG.ALTER.CFSTRUCT CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000042PERMIT MQZG.ALTER.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000043PERMIT MQZG.ALTER.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ1CHIN)
00000044PERMIT MQZG.ALTER.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ2CHIN)
00000045PERMIT MQZG.ALTER.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ3CHIN)
00000043PERMIT MQZG.ALTER.NAMELIST CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000044PERMIT MQZG.ALTER.PROCESS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000045PERMIT MQZG.ALTER.QALIAS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000046PERMIT MQZG.ALTER.QLOCAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000047PERMIT MQZG.ALTER.QMGR CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000048PERMIT MQZG.ALTER.QMODAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000049PERMIT MQZG.ALTER.QREMOTE CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000050PERMIT MQZG.ALTER.SECURITY CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000051PERMIT MQZG.ALTER.STGCLASS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000052PERMIT MQZG.ALTER.TRACE CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000053PERMIT MQZG.ARCHIVE.LOG CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000060PERMIT MQZG.BACKUP.CFSTRUCT CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000061PERMIT MQZG.CLEAR.QLOCAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000070PERMIT MQZG.DEFINE.AUTHINFO CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000071PERMIT MQZG.DEFINE.BUFFPOOL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000072PERMIT MQZG.DEFINE.CFSTRUCT CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000073PERMIT MQZG.DEFINE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000074PERMIT MQZG.DEFINE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ1CHIN)
00000075PERMIT MQZG.DEFINE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ2CHIN)
00000076PERMIT MQZG.DEFINE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ3CHIN)
00000074PERMIT MQZG.DEFINE.MAXSMSGS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000075PERMIT MQZG.DEFINE.NAMELIST CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000076PERMIT MQZG.DEFINE.PROCESS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000077PERMIT MQZG.DEFINE.PSID CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000078PERMIT MQZG.DEFINE.QALIAS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000079PERMIT MQZG.DEFINE.QLOCAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000080PERMIT MQZG.DEFINE.QMODEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000081PERMIT MQZG.DEFINE.REMOTE CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000082PERMIT MQZG.DEFINE.STGCLASS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000083PERMIT MQZG.DELETE.AUTHINFO CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000084PERMIT MQZG.DELETE.CFSTRUCT CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000085PERMIT MQZG.DELETE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000088PERMIT MQZG.DELETE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000089PERMIT MQZG.DELETE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ1CHIN)
00000090PERMIT MQZG.DELETE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ2CHIN)
00000091PERMIT MQZG.DELETE.CHANNEL CLASS(MQCMDS) ACC(ALTER) ID(MQZ3CHIN)
00000086PERMIT MQZG.DELETE.NAMELIST CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000087PERMIT MQZG.DELETE.PROCESS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000088PERMIT MQZG.DELETE.QALIAS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)

```

```

00000089PERMIT MQZG.DELETE.QLOCAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000090PERMIT MQZG.DELETE.QMODEL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000091PERMIT MQZG.DELETE.QREMOTE CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000092PERMIT MQZG.DELETE.STGCLASS CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000093PERMIT MQZG.DISPLAY.ARCHIVE CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000094PERMIT MQZG.DISPLAY.AUTHINFO CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000095PERMIT MQZG.DISPLAY.CFSTATUS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000096PERMIT MQZG.DISPLAY.CFSTRUCT CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000097PERMIT MQZG.DISPLAY.CHANNEL CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000104PERMIT MQZG.DISPLAY.CHANNEL CLASS(MQCMDS) ACC(READ) ID(MQZ1CHIN)
00000105PERMIT MQZG.DISPLAY.CHANNEL CLASS(MQCMDS) ACC(READ) ID(MQZ2CHIN)
00000106PERMIT MQZG.DISPLAY.CHANNEL CLASS(MQCMDS) ACC(READ) ID(MQZ3CHIN)
00000098PERMIT MQZG.DISPLAY.CHSTATUS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000108PERMIT MQZG.DISPLAY.CHSTATUS CLASS(MQCMDS) ACC(READ) ID(MQZ1CHIN)
00000109PERMIT MQZG.DISPLAY.CHSTATUS CLASS(MQCMDS) ACC(READ) ID(MQZ2CHIN)
00000110PERMIT MQZG.DISPLAY.CHSTATUS CLASS(MQCMDS) ACC(READ) ID(MQZ3CHIN)
00000111PERMIT MQZG.DISPLAY.CLUSQMGR CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000112PERMIT MQZG.DISPLAY.CLUSQMGR CLASS(MQCMDS) ACC(READ) ID(MQZ1CHIN)
00000113PERMIT MQZG.DISPLAY.CLUSQMGR CLASS(MQCMDS) ACC(READ) ID(MQZ2CHIN)
00000114PERMIT MQZG.DISPLAY.CLUSQMGR CLASS(MQCMDS) ACC(READ) ID(MQZ3CHIN)
00000100PERMIT MQZG.DISPLAY.CMDSERV CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000101PERMIT MQZG.DISPLAY.DQM CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000117PERMIT MQZG.DISPLAY.DQM CLASS(MQCMDS) ACC(READ) ID(MQZ1CHIN)
00000118PERMIT MQZG.DISPLAY.DQM CLASS(MQCMDS) ACC(READ) ID(MQZ2CHIN)
00000119PERMIT MQZG.DISPLAY.DQM CLASS(MQCMDS) ACC(READ) ID(MQZ3CHIN)
00000102PERMIT MQZG.DISPLAY.GROUP CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000103PERMIT MQZG.DISPLAY.LOG CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000104PERMIT MQZG.DISPLAY.MAXSMSGS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000105PERMIT MQZG.DISPLAY.NAMELIST CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000106PERMIT MQZG.DISPLAY.PROCESS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000107PERMIT MQZG.DISPLAY.QALIAS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000108PERMIT MQZG.DISPLAY.QCLUSTER CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000127PERMIT MQZG.DISPLAY.QCLUSTER CLASS(MQCMDS) ACC(READ) ID(MQZ1CHIN)
00000128PERMIT MQZG.DISPLAY.QCLUSTER CLASS(MQCMDS) ACC(READ) ID(MQZ2CHIN)
00000129PERMIT MQZG.DISPLAY.QCLUSTER CLASS(MQCMDS) ACC(READ) ID(MQZ3CHIN)
00000109PERMIT MQZG.DISPLAY.QLOCAL CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000110PERMIT MQZG.DISPLAY.QMGR CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000111PERMIT MQZG.DISPLAY.QMODEL CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000112PERMIT MQZG.DISPLAY.QREMOTE CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000113PERMIT MQZG.DISPLAY.QSTATUS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000114PERMIT MQZG.DISPLAY.QUEUE CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000115PERMIT MQZG.DISPLAY.THREAD CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000116PERMIT MQZG.DISPLAY.SECURITY CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000117PERMIT MQZG.DISPLAY.STGCLASS CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000118PERMIT MQZG.DISPLAY.SYSTEM CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000119PERMIT MQZG.DISPLAY.TRACE CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000120PERMIT MQZG.DISPLAY.USAGE CLASS(MQCMDS) ACC(READ) ID(SDRES1)
00000121PERMIT MQZG.MOVE.QLOCAL CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000130PERMIT MQZG.PING.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)

```

```

00000200PERMIT MQZG.RECOVER.BSDS CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000210PERMIT MQZG.RECOVER.CFSTRUCT CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000300PERMIT MQZG.REFRESH.CLUSTER CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000301PERMIT MQZG.REFRESH.CLUSTER CLASS(MQCMDS) ACC(ALTER) ID(MQZ1CHIN)
00000302PERMIT MQZG.REFRESH.CLUSTER CLASS(MQCMDS) ACC(ALTER) ID(MQZ2CHIN)
00000303PERMIT MQZG.REFRESH.CLUSTER CLASS(MQCMDS) ACC(ALTER) ID(MQZ3CHIN)
00000310PERMIT MQZG.REFRESH.QMGR CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000320PERMIT MQZG.REFRESH.SECURITY CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000400PERMIT MQZG.RESET.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000401PERMIT MQZG.RESET.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000402PERMIT MQZG.RESET.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000403PERMIT MQZG.RESET.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000410PERMIT MQZG.RESET.CLUSTER CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000411PERMIT MQZG.RESET.CLUSTER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000412PERMIT MQZG.RESET.CLUSTER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000413PERMIT MQZG.RESET.CLUSTER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000420PERMIT MQZG.RESET.QSTATS CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000430PERMIT MQZG.RESET.TPIPE CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000500PERMIT MQZG.RESOLVE.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000510PERMIT MQZG.RESOLVE.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000520PERMIT MQZG.RESOLVE.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000530PERMIT MQZG.RESOLVE.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000600PERMIT MQZG.RESUME.INDOUBT CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000700PERMIT MQZG.RVERIFY.SECURITY CLASS(MQCMDS) ACC(ALTER) ID(SDRES1)
00000800PERMIT MQZG.SET.ARCHIVE CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000810PERMIT MQZG.SET.LOG CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000820PERMIT MQZG.SET.SYSTEM CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000900PERMIT MQZG.START.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000901PERMIT MQZG.START.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000902PERMIT MQZG.START.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000903PERMIT MQZG.START.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000910PERMIT MQZG.START.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000911PERMIT MQZG.START.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000912PERMIT MQZG.START.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000913PERMIT MQZG.START.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000920PERMIT MQZG.START.CMDSERV CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000930PERMIT MQZG.START.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00000931PERMIT MQZG.START.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00000932PERMIT MQZG.START.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00000933PERMIT MQZG.START.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00000940PERMIT MQZG.START.TRACE CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001000PERMIT MQZG.STOP.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001001PERMIT MQZG.STOP.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00001002PERMIT MQZG.STOP.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00001003PERMIT MQZG.STOP.CHANNEL CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00001010PERMIT MQZG.STOP.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001011PERMIT MQZG.STOP.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00001012PERMIT MQZG.STOP.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00001013PERMIT MQZG.STOP.CHINIT CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)

```



```

00001020PERMIT MQZG.STOP.CMDSERV CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001030PERMIT MQZG.STOP.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001031PERMIT MQZG.STOP.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ1CHIN)
00001032PERMIT MQZG.STOP.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ2CHIN)
00001033PERMIT MQZG.STOP.LISTENER CLASS(MQCMDS) ACC(CONTROL) ID(MQZ3CHIN)
00001040PERMIT MQZG.STOP.QMGR CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001050PERMIT MQZG.STOP.TRACE CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)
00001100PERMIT MQZG.SUSPEND.QMGR CLASS(MQCMDS) ACC(CONTROL) ID(SDRES1)

```

Example: B-18 MQQUEUE class - Clist showing initial RACF definitions for the business scenario solution

```

00000010CONTROL ASIS
00000011RDELETE MQQUEUE MQZG.**
00000012RDELETE MQQUEUE MQZG.CSQ4SAMP.*
00000013RDELETE MQQUEUE MQZG.SYSTEM.*
00000020RDEFINE MQQUEUE MQZG.ACCOUNT.REQUEST.EC.CICS UACC(NONE)
00000021RDEFINE MQQUEUE MQZG.ATM.REQUEST.CC.CICS UACC(NONE)
00000022RDEFINE MQQUEUE MQZG.CICS01.INITQ UACC(NONE)
00000023RDEFINE MQQUEUE MQZG.DLQ.MQZ1 UACC(NONE)
00000024RDEFINE MQQUEUE MQZG.DLQ.MQZ2 UACC(NONE)
00000025RDEFINE MQQUEUE MQZG.DLQ.MQZ3 UACC(NONE)
00000026RDEFINE MQQUEUE MQZG.MQ400 UACC(NONE)
00000027RDEFINE MQQUEUE MQZG.MQ400.SC UACC(NONE)
00000028RDEFINE MQQUEUE MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED UACC(NONE)
00000029RDEFINE MQQUEUE MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC UACC(NONE)
00000030RDEFINE MQQUEUE MQZG.PORTFOLIO.BATCH.MQ400 UACC(NONE)
00000031RDEFINE MQQUEUE MQZG.PORTFOLIO.BATCH.MQ400.SC UACC(NONE)
00000032RDEFINE MQQUEUE MQZG.PORTFOLIO.REQUEST.EC.CICS UACC(NONE)
00000033RDEFINE MQQUEUE MQZG.SYSTEM.ADMIN.CHANNEL.EVENT UACC(NONE)
00000034RDEFINE MQQUEUE MQZG.SYSTEM.ADMIN.CONFIG.EVENT UACC(NONE)
00000035RDEFINE MQQUEUE MQZG.SYSTEM.ADMIN.PERFM.EVENT UACC(NONE)
00000036RDEFINE MQQUEUE MQZG.SYSTEM.ADMIN.QMGR.EVENT UACC(NONE)
00000037RDEFINE MQQUEUE MQZG.SYSTEM.CHANNEL.INITQ UACC(NONE)
00000038RDEFINE MQQUEUE MQZG.SYSTEM.CHANNEL.SYNCQ UACC(NONE)
00000039RDEFINE MQQUEUE MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE UACC(NONE)
00000040RDEFINE MQQUEUE MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE UACC(NONE)
00000041RDEFINE MQQUEUE MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE UACC(NONE)
00000042RDEFINE MQQUEUE MQZG.SYSTEM.COMMAND.INPUT UACC(NONE)
00000043RDEFINE MQQUEUE MQZG.SYSTEM.COMMAND.REPLY.MODEL UACC(NONE)
00000044RDEFINE MQQUEUE MQZG.SYSTEM.QSG.CHANNEL.SYNCQ UACC(NONE)
00000045RDEFINE MQQUEUE MQZG.SYSTEM.QSG.TRANSMIT.QUEUE UACC(NONE)
00000046RDEFINE MQQUEUE MQZG.SYSTEM.CSQOREXX.* UACC(NONE)
00000047RDEFINE MQQUEUE MQZG.SYSTEM.CSQUTIL.* UACC(NONE)
00000048RDEFINE MQQUEUE MQZG.SYSTEM.CSQXCMD.* UACC(NONE)
00000049RDEFINE MQQUEUE MQZG.CSQ4SAMP.* UACC(NONE)
00000050RDEFINE MQQUEUE MQZG.CSQ.* UACC(NONE)

```

Example: B-19 MQQUEUE E class - CLIST to grant initial access to limited user IDs to the profiles previously defined in Appendix B-18, "MQQUEUE class - Clist showing initial RACF definitions for the business scenario solution" on page 293

```

00000010CONTROL ASIS
00000020PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(ALTER)
ID(SDRES1)
00000021PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES5)
00000022PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000023PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000024PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000025PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)
00000026PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000027PERMIT MQZG.ACCOUNT.REQUEST.EC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000028PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES1)
00000029PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES4)
00000030PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES2)
00000031PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000032PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000033PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000034PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)
00000035PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000036PERMIT MQZG.ATM.REQUEST.CC.CICS CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000037PERMIT MQZG.CICS01.INITQ CLASS(MQQUEUE) ACC(NONE) ID(SDRES
00000038PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(ALTER) ID(SDRES1)
00000039PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES2)
00000040PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES3)
00000041PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES4)
00000042PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES5)
00000043PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1CHIN)
00000044PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2CHIN)
00000045PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3CHIN)
00000046PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1MSTR)
00000047PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2MSTR)

```

```

00000048PERMIT MQZG.DLQ.MQZ1 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3MSTR)
00000049PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(ALTER) ID(SDRES1)
00000050PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES2)
00000051PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES3)
00000052PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES4)
00000053PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES5)
00000054PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1CHIN)
00000055PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2CHIN)
00000056PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3CHIN)
00000057PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1MSTR)
00000058PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2MSTR)
00000059PERMIT MQZG.DLQ.MQZ2 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3MSTR)
00000060PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(ALTER) ID(SDRES1)
00000061PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES2)
00000062PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES3)
00000063PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES4)
00000064PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(SDRES5)
00000065PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1CHIN)
00000066PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2CHIN)
00000067PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3CHIN)
00000068PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ1MSTR)
00000069PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ2MSTR)
00000070PERMIT MQZG.DLQ.MQZ3 CLASS(MQQUEUE) ACC(CONTROL) ID(MQZ3MSTR)
00000071PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(NONE) ID(SDRES1)
00000072PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ1CHIN)
00000073PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ2CHIN)
00000074PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ3CHIN)
00000075PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ1MSTR)
00000076PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ2MSTR)
00000077PERMIT MQZG.MQ400 CLASS(MQQUEUE) ACC(ALTER) ID(MQZ3MSTR)
00000078PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(NONE) ID(SDRES1)
00000079PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ1CHIN)
00000080PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ2CHIN)
00000081PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ3CHIN)
00000082PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ1MSTR)
00000083PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ2MSTR)
00000084PERMIT MQZG.MQ400.SC CLASS(MQQUEUE) ACC(ALTER) ID(MQZ3MSTR)
00000085PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(ALTER) ID(SDRES1)
00000086PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(UPDATE) ID(SDRES4)
00000087PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ1CHIN)
00000088PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ2CHIN)
00000089PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ3CHIN)
00000090PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ1CHIN)

```

00000091PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ2CHIN)
 00000092PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ3CHIN)
 00000093PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(ALTER) ID(SDRES1)
 00000094PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(SDRES5)
 00000095PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ1CHIN)
 00000096PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ2CHIN)
 00000097PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ3CHIN)
 00000098PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ1MSTR)
 00000099PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ2MSTR)
 00000100PERMIT MQZG.PORTFOLIO.BATCH.COMPLETE.SHARED.SC CLASS(MQQUEUE)
 ACC(UPDATE) ID(MQZ3MSTR)
 00000101PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(ALTER)
 ID(SDRES1)
 00000102PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000103PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000104PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000105PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000106PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ2MSTR)
 00000107PERMIT MQZG.PORTFOLIO.BATCH.MQ400 CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ3MSTR)
 00000108PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(ALTER)
 ID(SDRES1)
 00000109PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000110PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000111PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000112PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000113PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ2MSTR)
 00000114PERMIT MQZG.PORTFOLIO.BATCH.MQ400.SC CLASS(MQQUEUE) ACC(UPDATE)
 ID(MQZ3MSTR)

00000115PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(ALTER)
 ID(SDRES1)
 00000116PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(SDRES5)
 00000117PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000118PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000119PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000120PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000121PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000122PERMIT MQZG.PORTFOLIO.REQUEST.EC.CICS CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000123PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000123PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000123PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000124PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000123PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000123PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000125PERMIT MQZG.SYSTEM.ADMIN.CHANNEL.EVENT CLASS(MQUEUE) ACC(ALTER)
 ID(SDRES1)
 00000129PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(ALTER)
 ID(SDRES1)
 00000130PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)
 00000131PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ2CHIN)
 00000132PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ3CHIN)
 00000133PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1MSTR)
 00000134PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ2MSTR)
 00000135PERMIT MQZG.SYSTEM.ADMIN.CONFIG.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ3MSTR)
 00000136PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(SDRES1)
 00000137PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQUEUE) ACC(UPDATE)
 ID(MQZ1CHIN)

```

00000138PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000139PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000140PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)
00000141PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000142PERMIT MQZG.SYSTEM.ADMIN.PERFM.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000143PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES1)
00000144PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000145PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000146PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000147PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)
00000148PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000149PERMIT MQZG.SYSTEM.ADMIN.QMGR.EVENT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(ALTER)
ID(SDRES1)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000150PERMIT MQZG.SYSTEM.CHANNEL.INITQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000151PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(ALTER)
ID(SDRES1)
00000151PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000150PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1MSTR)

```

```

00000150PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2MSTR)
00000150PERMIT MQZG.SYSTEM.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(SDRES1)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ2CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ3CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ2MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.COMMAND.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ3MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(SDRES1)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ1CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ2CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ3CHIN)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ1MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ2MSTR)
00000152PERMIT MQZG.SYSTEM.CLUSTER.REPOSITORY.QUEUE CLASS(MQQUEUE)
ACC(UPDATE) ID(MQZ3MSTR)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(SDRES1)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000154PERMIT MQZG.SYSTEM.CLUSTER.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(ALTER)
ID(MQZ1CHIN)
00000155PERMIT MQZG.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES1)

```

```

00000155PERMIT MQZG.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000155PERMIT MQZG.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000155PERMIT MQZG.SYSTEM.COMMAND.INPUT CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000156PERMIT MQZG.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ACC(UPDATE)
ID(SDRES1)
00000156PERMIT MQZG.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000156PERMIT MQZG.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000156PERMIT MQZG.SYSTEM.COMMAND.REPLY.MODEL CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000157PERMIT MQZG.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000157PERMIT MQZG.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000157PERMIT MQZG.SYSTEM.QSG.CHANNEL.SYNCQ CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000158PERMIT MQZG.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN)
00000158PERMIT MQZG.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ2CHIN)
00000158PERMIT MQZG.SYSTEM.QSG.TRANSMIT.QUEUE CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ3CHIN)
00000159PERMIT MQZG.SYSTEM.CSQOREXX.* CLASS(MQQUEUE) ACC(UPDATE) ID(SDRES1)
00000160PERMIT MQZG.SYSTEM.CSQUTIL.* CLASS(MQQUEUE) ACC(UPDATE) ID(SDRES1)
00000161PERMIT MQZG.SYSTEM.CSQXCMD.* CLASS(MQQUEUE) ACC(UPDATE)
ID(MQZ1CHIN,MQZ2CHIN,MQZ3CHIN)
00000170PERMIT MQZG.CSQ4SAMP.* CLASS(MQQUEUE) ACC(UPDATE) ID(SDRES1)
00000200PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(SDRES1)
00000300PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ1CHIN)
00000400PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ2CHIN)
00000500PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ3CHIN)
00000600PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ1MSTR)
00000700PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ2MSTR)
00000800PERMIT MQZG.CSQ.* CLASS(MQQUEUE) ACC(UPDATE) ID(MQZ3MSTR)

```

Once we had made the updates to the channel definitions to implement SSL, they looked like this:

```

CHANNELS:
ALTER CHANNEL(TO.CCLUS1.MQAIX2) +
    CHLTYPE(CLUSSDR) +
    QSGDISP(QMGR) +
    TRPTYPE(TCP) +
    SSLCIPH(NULL_SHA) +

```



```

        SSLPEER('0=Dolphin*')
ALTER CHANNEL(TO.ECLUS1.MQAX2) +
        CHLTYPE(CLUSSDR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('0=Dolphin*')

ALTER CHANNEL(TO.CCLUS1.MQZ1) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP)
        MCAUSER(SDRES4) +
        SSLCAUTH(REQUIRED) +
        SSLCIPH(NULL_SHA) +
        SSLPEER('0=Dolphin*')

DEFINE CHANNEL(TO.ECLUS1.MQZ1) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES1) +
        SSLCAUTH(REQUIRED) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('0=Dolphin*')

ALTER CHANNEL(TO.CCLUS1.MQZ2) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES4)
        SSLCAUTH(REQUIRED) +
        SSLCIPH(NULL_SHA) +
        SSLPEER('0=Dolphin*')

DEFINE CHANNEL(TO.ECLUS1.MQZ2) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES1) +
        SSLCAUTH(REQUIRED) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('0=Dolphin*')

ALTER CHANNEL(TO.CCLUS1.MQZ3) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES4) +

```

```

        SSLCAUTH(REQUIRED) +
        SSLCIPH(NULL_SHA) +
        SSLPEER('O=Dolphin*')

DEFINE CHANNEL(TO.ECLUS1.MQZ3) +
        CHLTYPE(CLUSRCVR) +
        QSGDISP(QMGR) +
        TRPTYPE(TCP)
        MCAUSER(SDRES1) +
        SSLCAUTH(REQUIRED) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('O=Dolphin*')

SENDER/RECEIVER TO MQ400
ALTER CHANNEL(MQZG.CC.MQ400) +
        CHLTYPE(SDR) +
        QSGDISP(GROUP) +
        TRPTYPE(TCP)

ALTER CHANNEL(MQZG.SC.MQ400) +
        CHLTYPE(SDR) +
        QSGDISP(GROUP) +
        TRPTYPE(TCP) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('O=SHETLAND*')

ALTER CHANNEL(MQ400.CC.MQZG) +
        CHLTYPE(RCVR) +
        QSGDISP(GROUP) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES4)

ALTER CHANNEL(MQ400.SC.MQZG) +
        CHLTYPE(RCVR) +
        QSGDISP(GROUP) +
        TRPTYPE(TCP) +
        MCAUSER(SDRES5) +
        SSLCIPH(TRIPLE_DES_SHA_US) +
        SSLPEER('O=SHETLAND*') +
        SSLCAUTH(REQUIRED)

```

OS/400

Object definitions

```
* QUEUE MANAGER:
ALTER QMGR +
      SSLKEYR ('/QIBM/UserData/mqm/qmgrs/MQ400/key')
```

```
* TRANSMISSION QUEUES:
DEFINE QLOCAL ('MQZG') +
      USAGE(XMITQ) +
      TRIGGER +
      TRIGTYPE(FIRST) +
      TRIGDPH(1) +
      INITQ('SYSTEM.CHANNEL.INITQ')
```

```
DEFINE QLOCAL ('MQZG.SC') +
      USAGE(XMITQ) +
      TRIGGER +
      TRIGTYPE(FIRST) +
      TRIGDPH(1) +
      INITQ('SYSTEM.CHANNEL.INITQ')
```

```
* LOCAL QUEUES:
DEFINE QLOCAL ('PORTFOLIO.BATCH') +
      TRIGGER +
      TRIGTYPE(FIRST) +
      PROCESS('PROCESS1') +
      INITQ('TRIGGER.QUEUE')
```

```
DEFINE QLOCAL ('PORTFOLIO.BATCH.SC') +
      TRIGGER +
      TRIGTYPE(FIRST) +
      PROCESS('PROCESS2') +
      INITQ('TRIGGER.QUEUE')
```

```
DEFINE QLOCAL ('TRIGGER.QUEUE')
```

```
* REMOTE QUEUES:
DEFINE QREMOTE ('PORTFOLIO.BATCH.COMPLETE') +
      XMITQ('MQZG') +
      RNAME('PORTFOLIO.BATCH.COMPLETE.SHARED') +
      RQMNAME('MQZG')
```

```
DEFINE QREMOTE ('PORTFOLIO.BATCH.COMPLETE.SC') +
      XMITQ('MQZG.SC') +
      RNAME('PORTFOLIO.BATCH.COMPLETE.SHARED.SC') +
      RQMNAME('MQZG')
```

```

* CHANNELS:
DEFINE CHANNEL ('MQ400.CC.MQZG') CHLTYPE(SDR) +
  CONNAME('9.12.6.106(1543)') +
  XMITQ('MQZG')

DEFINE CHANNEL ('MQ400.SC.MQZG') CHLTYPE(SDR) +
  CONNAME('9.12.8.11(1540)') +
  XMITQ('MQZG.SC') +
  SSLCIPH('TRIPLE_DES_SHA_US')

DEFINE CHANNEL ('MQZG.CC.MQ400') CHLTYPE(RCVR)

DEFINE CHANNEL ('MQZG.SC.MQ400') CHLTYPE(RCVR) +
  SSLCIPH('TRIPLE_DES_SHA_US')

* PROCESSES:
DEFINE PROCESS ('PROCESS1') REPLACE +
  APPLICID('JOHN/PROGRAM1')

DEFINE PROCESS ('PROCESS2') REPLACE +
  APPLICID('JOHN/PROGRAM2')

```

AIX

Listed below are script files for AIX.

Object definitions

Example: B-20 The following scripts show object definitions set up on the Qmgr MQAIX3 on pSeries server before changing the setup to take increased security into account.

```

** Object Definition for MQAIX3 (Before SSL)

** For full repository
DEFINE NAMELIST(DOLPHIN) NAMES(ECLUS1,CCLUS1,ECLUS2,CCLUS2)
ALTER QMGR REPOSNL(DOLPHIN)

** Cluster receiver channel
DEFINE CHANNEL(TO.CCLUS1.MQAIX3) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.175(1414)') +
  CLUSTER(CCLUS1)

DEFINE CHANNEL(TO.ECLUS1.MQAIX3) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.175(1414)') +

```

```

CLUSTER(ECLUS1)

DEFINE CHANNEL(TO.CCLUS2.MQAI3) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.175(1414)') +
  CLUSTER(CCLUS2)

DEFINE CHANNEL(TO.ECLUS2.MQAI3) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.175(1414)') +
  CLUSTER(ECLUS2)

** Cluster sender channel
DEFINE CHANNEL(TO.ECLUS1.MQAI2) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(ECLUS1)

DEFINE CHANNEL(TO.CCLUS1.MQAI2) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(CCLUS1)

DEFINE CHANNEL(TO.ECLUS2.MQAI2) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(ECLUS2)

DEFINE CHANNEL(TO.CCLUS2.MQAI2) +
  CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  CLUSTER(CCLUS2)

** Object Definition for MQAI3

** Queue Alias for request message
DEFINE QALIAS(ACCOUNT.REQUEST.EC) +
  TARGQ(ACCOUNT.REQUEST.EC.CICS) +
  DEFBIND(NOTFIXED) +
  CLUSTER(ECLUS2)

DEFINE QALIAS(PORTPOLIO.REQUEST.EC)+
  TARGQ(PORTPOLIO.REQUEST.EC.CICS) +
  DEFBIND(NOTFIXED) +
  CLUSTER(ECLUS2)

DEFINE QALIAS(ATM.REQUEST.CC) +
  TARGQ(ATM.REQUEST.CC.CICS) +
  DEFBIND(NOTFIXED) +

```

```

CLUSTER(CCLUS2)

** Remote Queue for response message
DEFINE QREMOTE(MQW2K2EC) +
  RQMNAME(MQW2K2ECB) +
  CLUSTER(ECLUS1)

DEFINE QREMOTE(MQW2K2CC) +
  RQMNAME(MQW2K2CCB) +
  CLUSTER(CCLUS1)

DEFINE QREMOTE(MQW2K5EC) +
  RQMNAME(MQW2K5ECB) +
  CLUSTER(ECLUS1)

DEFINE QREMOTE(MQW2K5CC) +
  RQMNAME(MQW2K5CCB) +
  CLUSTER(CCLUS1)

```

Example: B-21 The following scripts show object definitions set up on the Qmgr MQAIX2 on pSeries server after objects created and changed by security hardening.

```

** Object Definition for MQAIX2 (After SSL)

** For full repository
DEFINE NAMELIST(DOLPHIN) NAMES(ECLUS1,CCLUS1,ECLUS2,CCLUS2)

** For SSL
ALTER QMGR REPOSNL(DOLPHIN) +
  SSLKEYR=/var/mqm/qmgrs/MQAIX2/ssl/key

** Cluster receiver channel
DEFINE CHANNEL(TO.ECLUS2.MQAIX2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  SSLPEER('O=Dolphin*') +
  SSLCIPH(TRIPLE_DES_SHA_US) +
  CLUSTER(ECLUS2)

DEFINE CHANNEL(TO.CCLUS2.MQAIX2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
  CONNAME('9.20.18.71(1414)') +
  SSLPEER('O=Dolphin*') +
  SSLCIPH(NULL_SHA) +
  CLUSTER(CCLUS2)

DEFINE CHANNEL(TO.ECLUS1.MQAIX2) +
  CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +

```

```

        CONNAME('9.20.18.71(1414)')    +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(TRIPLE_DES_SHA_US)     +
        CLUSTER(ECLUS1)                +

DEFINE CHANNEL(TO.CCLUS1.MQAI2)      +
        CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +
        CONNAME('9.20.18.71(1414)')    +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(NULL_SHA)               +
        CLUSTER(CCLUS1)

** Cluster sender channel
DEFINE CHANNEL(TO.ECLUS2.MQAI3)      +
        CHLTYPE(CLUSSDR) TRPTYPE(TCP)  +
        CONNAME('9.20.18.175(1414)')   +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(TRIPLE_DES_SHA_US)     +
        CLUSTER(ECLUS2)

DEFINE CHANNEL(TO.CCLUS2.MQAI3)      +
        CHLTYPE(CLUSSDR) TRPTYPE(TCP)  +
        CONNAME('9.20.18.175(1414)')   +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(NULL_SHA)               +
        CLUSTER(CCLUS2)

DEFINE CHANNEL(TO.ECLUS1.MQAI3)      +
        CHLTYPE(CLUSSDR) TRPTYPE(TCP)  +
        CONNAME('9.20.18.175(1414)')   +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(TRIPLE_DES_SHA_US)     +
        CLUSTER(ECLUS1)

DEFINE CHANNEL(TO.CCLUS1.MQAI3)      +
        CHLTYPE(CLUSSDR) TRPTYPE(TCP)  +
        CONNAME('9.20.18.175(1414)')   +
        SSLPEER('0=Dolphin*')          +
        SSLCIPH(NULL_SHA)               +
        CLUSTER(CCLUS1)

** Queue Alias for request message
DEFINE QALIAS(ACCOUNT.REQUEST.EC)    +
        TARGQ(ACCOUNT.REQUEST.EC.CICS) +
        DEFBIND(NOTFIXED)             +
        CLUSTER(ECLUS2)

```

```

DEFINE QALIAS(PORTPOLIO.REQUEST.EC)  +
      TARGQ(PORTPOLIO.REQUEST.EC.CICS) +
      DEFBIND(NOTFIXED)                +
      CLUSTER(ECLUS2)

DEFINE QALIAS(ATM.REQUEST.CC)        +
      TARGQ(ATM.REQUEST.CC.CICS)      +
      DEFBIND(NOTFIXED)                +
      CLUSTER(CCLUS2)

** Remote Queue for response message

DEFINE QREMOTE(MQW2K2EC)  +
      RQMNAME(MQW2K2ECA)  +
      DEFBIND(NOTFIXED)  +
      CLUSTER(ECLUS1)

DEFINE QREMOTE(MQW2K2CC)  +
      RQMNAME(MQW2K2CCA)  +
      DEFBIND(NOTFIXED)  +
      CLUSTER(CCLUS1)

DEFINE QREMOTE(MQW2K5EC)  +
      RQMNAME(MQW2K5ECA)  +
      DEFBIND(NOTFIXED)  +
      CLUSTER(ECLUS1)

DEFINE QREMOTE(MQW2K5CC)  +
      RQMNAME(MQW2K5CCA)  +
      DEFBIND(NOTFIXED)  +
      CLUSTER(CCLUS1)

```

Windows

Listed below are script files for Windows 2000.

Object definitions

Example: B-22 The following scripts show object definitions set up on the Qmgr MQW2K1 on workstation1 before changing the setup to take increased security into account.

```

*LOCAL QUEUES Defined on MQW2K1:
DEFINE QLOCAL(ACCOUNT.RESPONSE.EC)
DEFINE QLOCAL(PORTFOLIO.RESPONSE.EC)
DEFINE QLOCAL(ATM.RESPONSE.CC)

```


*QMGR ALIAS Defined on MQW2K1:
*

```
DEFINE QREMOTE(MQW2K1ECA) +  
      RQMNAME(MQW2K1)      +  
      DEFBIND(NOTFIXED)   +  
      CLUSTER(ECLUS2)
```

```
DEFINE QREMOTE(MQW2K1ECB) +  
      RQMNAME(MQW2K1)      +  
      DEFBIND(NOTFIXED)   +  
      CLUSTER(ECLUS2)
```

```
DEFINE QREMOTE(MQW2K1CCA) +  
      RQMNAME(MQW2K1)      +  
      DEFBIND(NOTFIXED)   +  
      CLUSTER(CCLUS2)
```

```
DEFINE QREMOTE(MQW2K1CCB) +  
      RQMNAME(MQW2K1) +  
      DEFBIND(NOTFIXED) +  
      CLUSTER(CCLUS2)
```

RQMNAME(MQW2K1)

*CLUSTER CHANNELS defined on MQW2K1:
*

*To join CCLUS2
*

```
DEFINE CHANNEL(TO.CCLUS2.MQW2K1)+  
      CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +  
      CONNAME('9.20.24.241(1414)') +  
      CLUSTER(CCLUS2)
```

```
DEFINE CHANNEL(TO.CCLUS2.MQAI2)+  
      CHLTYPE(CLUSSDR) TRPTYPE(TCP) +  
      CONNAME('9.20.18.71(1414)') +  
      CLUSTER(CCLUS2)
```

* To join ECLUS2
*

```
DEFINE CHANNEL(TO.ECLUS2.MQW2K1)+  
      CHLTYPE(CLUSRCVR) TRPTYPE(TCP) +  
      CONNAME('9.20.24.241(1414)') +  
      CLUSTER(ECLUS2)
```

```
DEFINE CHANNEL(TO.ECLUS2.MQAI2)+  
      CHLTYPE(CLUSSDR) TRPTYPE(TCP) +
```

```

        CONNAME('9.20.18.71(1414)') +
    CLUSTER(ECLUS2)

* To setup client connection channel
DEFINE CHANNEL ('CLIENT.MQW2K1') CHLTYPE(SVRCONN) +
    MCAUSER('mquser1') +
    TRPTYPE(TCP) +
    REPLACE

* To setup the client connection table
DEFINE CHANNEL ('CLIENT.MQW2K2') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('itsoi.itso.hursley.ibm.com(1414)') +
    REPLACE

DEFINE CHANNEL ('CLIENT.MQW2K5') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('itsol.itso.hursley.ibm.com(1418)') +
    REPLACE

DEFINE CHANNEL ('CLIENT.MQW2K1') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('itsoo.itso.hursley.ibm.com(1414)') +
    REPLACE

```

Example: B-23 Object authority manager setup before security hardening.

```

@echo off
set QMGR=MQW2K5
setmqaut -m %QMGR% -t qmgr -g MQAPPS +connect
setmqaut -m %QMGR% -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t q -g MQAPPS -all
setmqaut -m %QMGR% -n **.RESPONSE.* -t q -g MQAPPS +get +browse

```

Example: B-24 Objects created and changed by security hardening. The example is from MQW2K5 on itsol.itso.hursley.ibm.com on port 1418

```

* -----
* Change channels
* -----
DEFINE CHANNEL ('TO.ECLUS2.MQAIX2') CHLTYPE(CLUSSDR) +
    TRPTYPE(TCP) +
    CLUSTER('ECLUS2') +
    CONNAME('9.20.18.71(1414)') +
    SSLCIPH('TRIPLE_DES_SHA_US') +
    SSLPEER('O=Dolphin Bank*') +
    REPLACE

DEFINE CHANNEL ('TO.ECLUS2.MQAIX3') CHLTYPE(CLUSSDR) +
    TRPTYPE(TCP) +
    CLUSTER('ECLUS2') +
    CONNAME('itsoaix3(1414)') +

```

```

        SSLCIPH('TRIPLE_DES_SHA_US') +
        SSLPEER('O=Dolphin Bank*') +
        REPLACE

DEFINE CHANNEL ('TO.ECLUS2.MQW2K5') CHLTYPE(CLUSRCVR) +
        TRPTYPE(TCP) +
        CLUSTER('ECLUS2') +
        CONNAME('9.20.14.178(1418)') +
        SSLCAUTH(REQUIRED) +
        SSLCIPH('TRIPLE_DES_SHA_US') +
        SSLPEER('O=Dolphin Bank*') +
        REPLACE

* -----
* Setup local qaliases to cluster queues
* -----
DEFINE QALIAS ('ACCOUNT.REQUEST.EC.LOCAL') +
        DESCR(' ') +
        PUT(ENABLED) +
        DEFPRTY(0) +
        DEFPSIST(NO) +
        SCOPE(QMGR) +
        GET(ENABLED) +
        TARGQ('ACCOUNT.REQUEST.EC') +
        CLUSTER(' ') +
        CLUSNL(' ') +
        DEFBIND(OPEN) +
        REPLACE

DEFINE QALIAS ('ATM.REQUEST.CC.LOCAL') +
        DESCR(' ') +
        PUT(ENABLED) +
        DEFPRTY(0) +
        DEFPSIST(NO) +
        SCOPE(QMGR) +
        GET(ENABLED) +
        TARGQ('ATM.REQUEST.CC') +
        CLUSTER(' ') +
        CLUSNL(' ') +
        DEFBIND(OPEN) +
        REPLACE

DEFINE QALIAS ('PORTFOLIO.REQUEST.EC.LOCAL') +
        DESCR(' ') +
        PUT(ENABLED) +
        DEFPRTY(0) +
        DEFPSIST(NO) +
        SCOPE(QMGR) +
        GET(ENABLED) +
        TARGQ('PORTFOLIO.REQUEST.EC') +
        CLUSTER(' ') +

```

```

    CLUSNL(' ') +
    DEFBIND(OPEN) +
    REPLACE
* -----
*Update client channels and client connection table
* -----
DEFINE CHANNEL ('CLIENT.MQW2K5') CHLTYPE(SVRCONN) +
    MCAUSER('mquser1') +
    SSLCAUTH(REQUIRED) +
    SSLCIPH('TRIPLE_DES_SHA_US') +
    SSLPEER('O=Dolphin Bank*') +
    REPLACE

DEFINE CHANNEL ('CLIENT.MQW2K1') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('itsoo.itso.hursley.ibm.com(1414)') +
    QMNAME('MQW2K1') +
    SSLCIPH('TRIPLE_DES_SHA_US') +
    SSLPEER('O=Dolphin Bank*') +
    REPLACE

DEFINE CHANNEL ('CLIENT.MQW2K2') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('9.20.31.89(1414)') +
    SSLCIPH('TRIPLE_DES_SHA_US') +
    SSLPEER('O=Dolphin Bank*') +
    REPLACE

DEFINE CHANNEL ('CLIENT.MQW2K5') CHLTYPE(CLNTCONN) +
    TRPTYPE(TCP) +
    CONNAME('9.20.30.95(1418)') +
    SSLCIPH('TRIPLE_DES_SHA_US') +
    SSLPEER('O=Dolphin Bank*') +
    REPLACE

```

Example: B-25 OAM Security script after security hardening

```

@echo off
set QMGR=MQW2K5
REM -- REMOVE PERMISSIONS PREVIOUSLY SET TO CLUSTER OBJECTS --
    setmqaut -m %QMGR% -n SYSTEM.CLUSTER.TRANSMIT.QUEUE -t q -g MQAPPS -all

REM -- Give rights to connect to the queue manager --
    setmqaut -m %QMGR% -t qmgr -g MQAPPS +connect

REM -- Give rights to get from response queues --
    setmqaut -m %QMGR% -n **.RESPONSE.* -t q -g MQAPPS +get +browse

REM -- Give rights to new local alias objects --
    setmqaut -m %QMGR% -n **.REQUEST.* -t q -g MQAPPS +put

```

JCL and code by platform

This section contains JCL and code used for our business scenario listed by platform.

z/OS

The JCL below is an example of using **CSQUTIL** with the command **MAKEDEF**.

It makes **DEFINE** commands for all the objects output by the **DISPLAY** commands included.

The data set named must exist before running this job and should be RECFM=FB, LRECL=80. This can then be used as an input for a later invocation of the command function or it can be incorporated into the initialization data sets CSQINP1 and CSQINP2.

You should check the output file and remove any definitions you do not wish to have there or tailor the **DISPLAY** command so you only get the objects you want.

For more information about **CSQUTIL** and **MAKEDEF** see Chapter 17, “WebSphere MQ utility program(CSQUTIL)”, in *WebSphere MQ for z/OS System Administration Guide Version 5 Release 3, SC34-6053*.

Example: B-26 JCL using CSQUTIL with the command MAKEDEF.

```
//CSQUTIL1 JOB MQSERIES, 'MQSERIES', TIME=1439,
//          CLASS=A, MSGCLASS=T, MSGLEVEL=(1,1),
//          NOTIFY=&SYSUID, REGION=0M
/*JOBPARM SYSAFF=SC61
//COMMAND EXEC PGM=CSQUTIL, PARM='MQZ1'
//STEPLIB DD DISP=SHR, DSN=MQ530.SCSQANLE
//          DD DISP=SHR, DSN=MQ530.SCSQAUTH
//CMDBK DD DSN=MQZ1.BACKUP.OBJ, DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
          COMMAND MAKEDEF(CMDBK)
/*
//CSQUCMD DD *
DISPLAY CHANNEL(*) ALL
DISPLAY QUEUE(*) ALL
DISPLAY PROCESS(*) ALL
DISPLAY NAMLIST(*) ALL
DISPLAY STGCLASS(*) ALL
/*
```

OS/400

PROGRAM1&2 referred to in Appendix 8, “Business scenario architecture” on page 141 is a simple example of how CL can be used to batch together commands to be triggered on receiving a trigger event.

```
Columns . . . : 1 71          Edit          JOHN/QCLSRC
SEU=>
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
***** Beginning of data *****
0001.00 PGM          PARM(&PARAM)
0002.00 DCL          VAR(&PARAM) TYPE(*CHAR) LEN(684)
0003.00 CALL PGM(QMQM/AMQSGET4) PARM(PORTFOLIO.BATCH MQ400)
0004.00 CALL PGM(QMQM/AMQSPUT4) PARM(RESPONSE MQ400)
0005.00 ENDPGM
***** End of data *****

F3=Exit   F4=Prompt   F5=Refresh   F9=Retrieve   F10=Cursor   F11=Toggle
F16=Repeat find   F17=Repeat change   F24=More keys
(C) COPYRIGHT IBM CORP. 1981. 2000.
```

Figure 11-64 PROGRAM1 - CL program calling sample get and put programs

In the figure above, it is necessary to declare the parameters that come with a trigger message and its length in lines 1 and 2. The sample AMQSGET4 program is then called to remove the messages from the queue. After the default get timeout has expired, the sample AMQSPUT4 program is called with an input file called RESPONSE. The first line of the input file is the queue name and the subsequent lines are the messages.

Contents of our response file:

```
PORTFOLIO.BATCH.COMPLETE
FILE RECEIVED
```

The commands used to create the program were as follows:

```
CRTLIB JOHN
CRTSRCPF FILE(JOHN/QCLSRC)
STRSEU SRCFILE(JOHN/QCLSRC) SRCMBR(PROGRAM1) TYPE(CLP)
CRTCLPGM PGM(JOHN/PROGRAM1) SRCFILE(JOHN/QCLSRC) SRCMBR(PROGRAM1)
```

Sample code

This section contains various program sources used in the setup for our business scenario.

Program source for SSL

MQ Client Program using MQCONN.

```
#include <stdio.h>
#include <string.h>
/*****
/* Samle Program for MQ Client with SSL (sslput.c) */
/*****
/* "sslput" write the messages in the "inpfile". */
/* */
/*****
#include <stdlib.h>
#include <cmqc.h>
#include <cmqxc.h>

main(int argc, char *argv[]){
    FILE *fp;
    char buffer[100];
    MQLONG i=0;
    MQOD mqod = {MQOD_DEFAULT};
    MQMD mqmd = {MQMD_DEFAULT};
    MQPMO pmo = {MQPMO_DEFAULT};
    MQCNO connect_options = {MQCNO_DEFAULT};
    MQCD mycd = {MQCD_CLIENT_CONN_DEFAULT};
    MQSCO mysco = {MQSCO_DEFAULT};
    MQCHAR qmname[MQ_Q_MGR_NAME_LENGTH];
    MQHCONN Hcon;
    MQHOBJ Hobj;
    MQLONG open_options;
    MQLONG close_options;
    MQLONG compcode;
    MQLONG reason;
    int cc;
    int count;

    if(argc!=5){
        printf("Usage: sslputc QName QmName ConnName Message-Count");
        exit(1);
    }
}
```

```

count = atoi(argv[4]);
strncpy(qmname, argv[2], MQ_Q_MGR_NAME_LENGTH);
strncpy(mycd.ConnectionName, argv[3], MQ_CONN_NAME_LENGTH);
strncpy(mycd.ChannelName, "SSL.CLCHL", MQ_CHANNEL_NAME_LENGTH);
printf("Channel Name  :%s\n", mycd.ChannelName);

/* For SSL */
/** Specify Keystore file name without ".sto" */
strncpy(mysco.KeyRepository, "C:\\SSL\\mykey",
MQ_SSL_KEY_REPOSITORY_LENGTH);
printf("Key Repository:%s\n", mysco.KeyRepository);

/** Specify CipherSpec */
strcpy(mycd.SSLCipherSpec, "TRIPLE_DES_SHA_US");
connect_options.SSLConfigPtr = &mysco;
connect_options.ClientConnPtr=&mycd;
connect_options.Version = MQCNO_VERSION_4;
printf("CipherSpec   :%s\n", mycd.SSLCipherSpec);

/* Connect Qmgr */
MQCONNX(qmname, &connect_options, &Hcon, &compcode, &reason);
if (compcode == MQCC_FAILED){
printf("MQCONNX ended with reason code %ld\n", reason);
exit( (int)reason );
}
strcpy(mqod.ObjectName, argv[1]);

/* Open a Queue */
open_options = MQOO_OUTPUT + MQOO_FAIL_IF QUIESCING;
printf("\nOpening the queue %s\n", mqod.ObjectName);
MQOPEN(Hcon, &mqod, open_options, &Hobj, &compcode, &reason);
if(reason!=MQRC_NONE){
printf("MQOPEN failed with reason code %ld\n", reason);
exit((int)reason);
}

if (compcode == MQCC_FAILED){
printf("unable to open queue for output\n");
}

/* Open the "inpfile" and get messages */
if((fp=fopen("inpfile", "r"))==NULL){
printf("Cannot read file: inpfile \n");
exit(1);
}
while((buffer[i]=fgetc(fp))!=EOF){
printf("%c ",buffer[i]);
i++;
}

```



```

    }

/* Set MQMD */
memcpy(mqmd.Format, MQFMT_STRING, (unsigned int)MQ_FORMAT_LENGTH);
mqmd.Persistence = MQPER_PERSISTENT;
memcpy(mqmd.MsgId, MQMI_NONE, sizeof(mqmd.MsgId));
strcpy(mqmd.ReplyToQ, "REPORTQ");

/* Put the message */
for(cc=0; cc<count; cc++){
    MQPUT(Hcon, Hobj, &mqmd, &pmo, i, buffer, &compcode, &reason);
    if (reason != MQRC_NONE){
        printf("MQPUT ended with reason code %ld\n", reason);
    }
}
fclose(fp);

/* Close the queue */
close_options = 0 ;
MQCLOSE(Hcon, &Hobj, close_options, &compcode, &reason);
printf("\n%d PUT successful!! Disconnecting from QMgr...", count);

/* Disconnect QMgr */
MQDISC(&Hcon, &compcode, &reason);
}

```

Sample code on z/OS

This section contains program source for z/OS.

Assembler program source

```

TITLE 'MQVPSTOQ - Load a PS file to a Queue'
        SPACE 2
* =====
* =
* =          MQVPSTOQ
* =          =====
* =
* = Copy a Sequential file into a Queue, with a zero length
* = record as end-of-file marker.
* =
* = An empty source file will result in a single zero-length
* = record being placed in the Queue.

```

```

* =
* =
* = Description
* = =====
* =
* = This program will copy the contents of the SEQUENTIAL file
* = in //SYSUT1 to the queue named in //SYSIN.
* =
* = The last message in the Queue is indicated by a zero-length
* = record. Applications MUST read this zero-length message
* = from the Queue to have processed all the records loaded
* = from the file.
* =
* =
* =
* =====
* SPACE 2
* EJECT
* SPACE 2
* =====
* =
* =
* = Register Usage
* = =====
* =
* = R0 ->
* = R1 -> Parmlists
* = R2 ->
* = R3 ->
* = R4 ->
* = R5 ->
* = R6 ->
* = R7 -> S/R Linkage
* = R8 ->
* = R9 ->
* = R10 -> Base 1
* = R11 -> Base 2
* = R12 -> Base 3
* = R13 -> Save Area
* = R14 ->
* = R15 ->
* =
* =
* = Return Codes
* = =====
* =
* = 0 -> The loading succeeded
* = 4 -> The Maximum Record Size for the Queue is too
* = small for the file
* = 8 -> A MQPUT to the Queue failed

```

```

* =      12 -> A MQINQ to process the queue Message           =
* =          maximum size failed OR the COMMIT operation     =
* =          to 'force' the records to the Queue failed      =
* =      16 -> The MQCONN to the requested Queue Manager failed =
* =          or the MQOPEN to the required Queue failed      =
* =          or the MQCLOSE for the required Queue failed    =
* =
* =      JCL                                                 =
* =      ===                                               =
* =
* =      //          EXEC PGM=MQVPSTOQ,PARM='<Queue Manager name>' =
* =      //STEPLIB DD DSN=                                     =
* =      //SYSPRINT DD SYSOUT=*,                             =
* =      //          DCB=(DSORG=PS,RECFM=VBA,LRECL=133,BLKSIZE=137) =
* =      //SYSIN    DD *,                                     =
* =      //          DCB=(DSORG=PS,RECFM=F,LRECL=80,BLKSIZE=80) =
* =      <queue to process>                                  =
* =      /*                                                 =
* =      //SYSUT1   DD DSN=<sequential file to load into the queue> =
* =      /*                                                 =
* =
* =
* =      Messages in SYSPRINT                               =
* =      =====                                           =
* =
* =
* =      1) END OF FILE DETECTED ON SYSIN - NO QUEUE NAME SUPPLIED =
* =
* =          The SYSIN file names the Queue to be loaded, but =
* =          there was no record in the file.                 =
* =          Processing stops.                                =
* =
* =
* =      2) MQCONNECT TO QM <Queue Manager Name> FAILED WITH CC <cc> =
* =          AND RC <rc>                                     =
* =
* =          The Queue Manager named in the parameter (or the =
* =          local Queue Manager if this did not exist or    =
* =          started with X'00' or X'40') is not connectable. =
* =          Processing stops.                                =
* =
* =
* =      3) MQOPEN FOR QUEUE <queue name> FAILED WITH CC <cc> AND =
* =          RC <rc>                                         =
* =
* =          Usage of the named Queue (which is to be loaded) =
* =          failed.                                         =
* =          Processing stops.                                =

```

```

* =
* =
* = 4) MQINQ FOR QUEUE <queue name> FAILED WITH CC <cc> AND
* = RC <rc>
* =
* = MQINQ is used to determine the Maximum Record
* = Length for the Queue to be loaded, but this
* = operation failed.
* = Processing stops.
* =
* =
* = 5) FILE LRECL OF <length> > QUEUE <queue name> MAX MESSAGE
* = LEN OF <length>
* =
* = The Files records will not fit into the Queue
* = without loosing data.
* = Processing stops.
* =
* =
* = 6) MQPUT FOR QUEUE <queue name> FAILED WITH CC <cc> AND
* = RC <rc>
* =
* = An attempt to add the record from the file
* = into the Queue failed.
* = Processing stops.
* =
* =
* = 7) LAST MQPUT FOR QUEUE <queue name> FAILED WITH CC <cc> AND
* = RC <rc>
* =
* = The writing of the last zero-length message
* = into the Queue failed.
* = Processing stops.
* =
* =
* = 8) COMMIT PROCESSING ON QUEUE <queue name> FAILED WITH
* = CC <cc> AND RC <rc>
* =
* = The MQ Syncpoint operation (which makes
* = the added records into the queue usable)
* = failed, and so the additions were
* = removed.
* = Processing stops.
* =
* =
* = 9) MQCLOSE FOR QUEUE <queue name> FAILED WITH CC <cc> AND
* = RC <rc>
* =
* = An attempt to Close the named Queue failed
* =

```

```

* =                Processing stops.                =
* =                                                    =
* =                                                    =
* = 10) <n> RECORDS LOADED INTO QUEUE <queue name>    =
* =                                                    =
* =                This numbers of records (including the last =
* =                zero-length one) were added to the queue, and =
* =                were committed there.                =
* =                Processing continues (end of Program)    =
* =                                                    =
* =                In all cases, the Completion Codes and the Return Codes =
* =                are as described in the MQM documentation. =
* =                                                    =
* =                                                    =
* =                                                    =
* = Failures                                           =
* = =====                                           =
* =                                                    =
* = As this loading is under Syncpoint control, a failure will =
* = leave the to-be-written queue unchanged. Consequently, =
* = the step can be rerun without any problems.          =
* =                                                    =
* =                                                    =
* = Linkage Editor                                     =
* = =====                                           =
* =                                                    =
* = To link this program, you must use AMODE(24)       =
* = and RMODE(24). The program is not designed for     =
* = multi-thread or re-entrant usage. The              =
* = IBM-supplied CSQBSTUB must be INCLUDED as well.    =
* =                                                    =
* = The entry point name is VPSTOQ.                   =
* =                                                    =
* = INCLUDE SYSLIB(CSQBSTUB)                           =
* = ENTRY VPSTOQ                                       =
* = MODE AMODE(24),RMODE(24)                           =
* = NAME MQVPSTOQ(R)                                   =
* =                                                    =
* =                                                    =
* = Customisation                                     =
* = =====                                           =
* =                                                    =
* = None needed.                                       =
* =                                                    =
* = You can invoke this module from within a program by =
* = setting R1 to point to a JCL type of parameter    =
* = (R1 -> Address -> LL....parm...), R13 to the      =
* = Save area, R14 to the Return Address, and R15 to   =

```

```

* =          the entry point before Branching to R15. Note that          =
* =          this module MUST be entered in 24-bit mode, and be        =
* =          loaded below the line.                                     =
* =                                                                 =
* =                                                                 =
* =          Copyright and Support                                     =
* =          =====                                               =
* =                                                                 =
* =          This program is not supported by any way by IBM. It is    =
* =          distributed only to show techniques for MQ usage. It may  =
* =          be freely modified and adapted by your installation.     =
* =                                                                 =
* =                                                                 =
* =          Author:  Robert Harris,                                  =
* =                   Message Queueing/Performance,                 =
* =                   IBM Hursley Park,                               =
* =                   England.                                       =
* =                                                                 =
* =          Lighty Modified: Peter Rhys Jenkins 2002 for SA-A227-R Redbook =
* =                   For ITSO,                                       =
* =                   IBM Hursley Park,                               =
* =                   United Kingdom.                                 =
* =                                                                 =
* =          Copyright IBM UK Labs Ltd, 1994,2002                    =
* =                                                                 =
* =                                                                 =
* =          =====                                               =
* =                   SPACE 2                                         =
* =                   EJECT                                           =
* =                   SPACE 2                                         =
* =          =====                                               =
* =                                                                 =
* =          Logic                                                  =
* =          =====                                               =
* =                                                                 =
* =          The program loads a queue by issing the apt MQPUT commands. =
* =                                                                 =
* =          After getting addressibility, the parameter is obtained.  =
* =                   This contains the name of the Queue Manager to create =
* =                   the queues within. If this parameter does not exist, =
* =                   or starts with a null or a blank, then the default =
* =                   Queue Manager will be used.                    =
* =                                                                 =
* =          The //SYSIN file is then read to obtain the name of the  =
* =                   queue to be loaded.                             =
* =                                                                 =
* =          The //SYSUT1 data file (which is to be loaded into the  =

```

```

* =      Queue) is then opened, and the maximum record length      =
* =      obtained.                                                =
* =                                                                =
* =      Contact is then made to the given Queue Manager, and the  =
* =      required Queue MQOPENed for access.                       =
* =                                                                =
* =      In order to obtain the Maximun Permitted Message Length   =
* =      for the Queue (used to trap a load which would result    =
* =      in data loss), a MQINQ is then performed.                =
* =      If the Queues Message size is too small for the file,    =
* =      then an error message is issued, and processing stops.   =
* =                                                                =
* =      A loop is then entered wherein a record is read from the  =
* =      file and MQPUTted into the Queue. Upon End of File,     =
* =      a zero-length message is written to the Queue. This     =
* =      is done so that future readers (which may be using      =
* =      MQGETS with waits) can detect when no more messages    =
* =      are present).                                           =
* =                                                                =
* =      After successfully writing the zero-length record to the   =
* =      Queue, a MQ Syncpoint is issued to commit all the       =
* =      loaded records to the queue. Until this operation        =
* =      succeeds, the records are not 'present' in the queue,   =
* =      so any sort of failure will leave the Queue in its     =
* =      pristine condition - thus permitting a simple rerun    =
* =      to perform the loading.                                  =
* =                                                                =
* =      After this, then all files are closed, the Queue closed, =
* =      and contact with the Queue Manager terminated.          =
* =                                                                =
* =      Note that if this step ends without doing a Syncpoint or a =
* =      Rollback, then an implicit Syncpoint is taken.         =
* =                                                                =
* =      =====
* =      SPACE 2
* =      EJECT
* =      SPACE 2
* =      =====
* =
* =      Standard MQ DSECTS
* =
* =      =====
* =      SPACE 2
* =      PRINT ON
* =      PRINT ON
* =      CMQA EQUONLY=YES,LIST=YES
* =      PRINT OFF
* =      SPACE 2

```

```

SPACE 2
PRINT ON
SPACE 2
EJECT
SPACE 2
* =====
* =
* =          DCB definition          =
* =
* =====

SPACE 2
DCBD DSORG=(BS,DA),DEV=(DA)
SPACE 2
EJECT
SPACE 2
* =====
* =
* =          Register definitions    =
* =
* =====

SPACE 2
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
EJECT
SPACE 2
* =====
* =
* =          Start of Program CSECT  =
* =
* =====

SPACE 2
* =====> <=====
* =====> Note the AMODE(24) and RMODE(24) settings. <=====
* =====> <=====
SPACE 2

```



```

MQVPSTOQ CSECT
MQVPSTOQ AMODE 24
MQVPSTOQ RMODE 24
        SPACE 2
        DC    CL8 '*RAHRAH*'
        DC    CL8 'MQVPSTOQ'
        DC    CL8 '&SYSDATE'
        DC    CL8 '&SYSTIME'
        SPACE 2
VPSTOQ  DS    0H                                Entry Point
        SPACE 1
        ENTRY VPSTOQ
        SPACE 2
        USING *,R15
        SPACE 1
        EJECT
        SPACE 2
* =====
* =
* =          Save all Registers
* =
* =
* =====
        SPACE 2
        STM   R14,R12,12(R13)                   Save Regs in higher SA
        STM   R0,R15,REGSSAVE                   Save in my Save Area
        B     INIT
        SPACE 2
REGSSAVE DS    17F
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =          Get local Addressability
* =
* =
* =====
        SPACE 2
INIT    DS    0H
        SPACE 1
* =====> <=====
* =====> Base registers are R10,R11,R12 (due to large <=====
* =====> usage of storage by the MQ Descriptor and <=====
* =====> Parameter macros). <=====
* =====> <=====
        SPACE 1
        BASR R10,0                               Set Base Reg
        USING *,R10,R11,R12                     Base here
        LA   R11,4094(,R10)                     Set 2nd Base reg
        LA   R11,2(,R11)

```

```

LA    R12,4094(,R11)          Set 3rd Base reg
LA    R12,2(,R12)
SPACE 2
STM   R0,R15,SAVE1           Save all Registers
ST    R13,SAVEOS1+4          Chain OS-type Savearea
LR    R15,R13                 Save higher SA address
LA    R13,SAVEOS1            Set new OS SA address
ST    R13,8(R15)             Chain OS-type Save area
SPACE 2
EJECT
SPACE 2
* =====
* =
* =          Get the Parm (Local Queue Manager name)
* =
* =====
SPACE 2
GETPARM DS 0H
SPACE 2
* =====> <=====
* =====> R1, the JCL Parameter, contains the name of <=====
* =====> the required Queue Manager for usage. Omitting <=====
* =====> the Parameter, causes the default LOCAL Queue <=====
* =====> Manager to be used. <=====
* =====> <=====
SPACE 1
ST    R1,PARMADDR            Save Parm address
SPACE 1
C     R1,=F'0'                Any parm?
BE    LOCALQM                 No - local QM
CLC   0(4,R1),=F'0'           Any parm?
BE    LOCALQM                 No - local QM
L     R2,0(,R1)               Address Parm area
CLC   0(2,R2),=H'0'           Any parm?
BE    LOCALQM                 No - local QM
*                                     Yes - parm supplied
LH    R14,0(,R2)              Get parm length
LA    R15,2(,R2)              Point to parm data
CLI   0(R15),X'00'            Any char?
BE    LOCALQM                 No - local QM
CLI   0(R15),X'40'            Any name?
BE    LOCALQM                 No - local QM
B     NAMEDQM                 Yes - named QM
SPACE 1
NAMEDQM DS 0H
SPACE 1
* =====> <=====
* =====> The maximum permitted length of the Queue <=====
* =====> Manager is checked, and the supplied name <=====

```

```

*      =====> truncated if necessary (this will probably      <=====
*      =====> lead to an inability to issue verbs to a      <=====
*      =====> non-existent QM later on!)                      <=====
*      =====>                                              <=====
SPACE 1
C      R14,=AL4(MAXQML)                Name too long?
BH     NQM1                            Yes - shorten it
B      NQM2                            No - OK
SPACE 1
NQM1  DS  0H
SPACE 1
L      R14,=AL4(MAXQML)                Set Max QM nam length
B      NQM2
SPACE 1
NQM2  DS  0H
SPACE 1
SH     R14,='H'1'                      Into offset
EX     R14,NQM1                        Copy QM name
B      GETPARMX
SPACE 1
NQM1  MVC  REQQM(0),0(R15)              EX - parm->localQM name
SPACE 1
LOCALQM DS  0H
SPACE 1
MVC   REQQM,REQQML                    Set local QM wanted
B     GETPARMX
SPACE 1
GETPARMX DS  0H
SPACE 2
EJECT
SPACE 2
* =====
* =
* =          Open the SYSPRINT DCB
* =
* =====
SPACE 2
*      =====>                                              <=====
*      =====> The SYSPRINT DDNAME is used to indicate      <=====
*      =====> info and errors. It is required.              <=====
*      =====>                                              <=====
SPACE 1
OPENSYSP DS  0H
SPACE 1
OPEN  (SYSPRINT,(OUTPUT)),MODE=24
SPACE 2
EJECT
SPACE 2
* =====

```

```

* =
* =          Process the SYSIN DCB          =
* =
* =====
      SPACE 2
DOSYSIN DS  0H
      SPACE 2
*      =====>                               <=====
*      =====> The SYSIN  DDNAME is used to indicate the <=====
*      =====> Queue to be loaded. It is required.         <=====
*      =====>                               <=====
      SPACE 1
SYSINO  DS  0H
      SPACE 1
      OPEN (SYSIN,(INPUT)),MODE=24
      SPACE 1
*      =====>                               <=====
*      =====> The SYSIN file is read to obtain a Queue name <=====
*      =====> Note that End-of-File is an error, and is    <=====
*      =====> processed accordingly.                         <=====
*      =====>                               <=====
      SPACE 1
SYSINR  DS  0H
      SPACE 1
      MVC  THISCOMM,=CL40'READ'
      XC   QSREC,QSREC          Clear SYSIN Buffer
      SPACE 1
      GET  SYSIN,QSREC
      SPACE 1
      MVC  WANTQAM,QSREC       Get the Queue Name
      SPACE 2
*      =====>                               <=====
*      =====> Close the SYSIN file                          <=====
*      =====>                               <=====
      SPACE 1
      CLOSE SYSIN
      SPACE 2
      B    OPSEQ              and open data file
      SPACE 1
DOSYSINX DS  0H
      SPACE 2
      EJECT
      SPACE 2
* =====
* =
* =          SYSIN processing - End of file          =
* =
* =====
      SPACE 2

```

```

*          =====>                                     <=====
*          =====> This routine is entered via the DCB when the <=====
*          =====> SYSIN file hits End-of-File. It shows an <=====
*          =====> error condition (no queue name supplied). <=====
*          =====>                                     <=====
*          =====> Print the error, and end processing <=====
*          =====>                                     <=====
SPACE 2
SYSINE DS 0H
SPACE 1
MVC PLINE1,MSG1          Say EOF detected
BAL R7,PRINT
SPACE 1
B EXIT                  End of Q deletions
SPACE 2
SPACE 2
EJECT
SPACE 2

* =====
* =
* =          Open the data file          =
* =
* =====

SPACE 2
*          =====>                                     <=====
*          =====> The SYSUT1 file contains the data to be loaded.<=====
*          =====> When opening the file, the LRECL is obtained <=====
*          =====> so that one can ensure that the queue wherein <=====
*          =====> these records are bound has a big enough <=====
*          =====> message length to take the records. <=====
*          =====>                                     <=====
SPACE 1
OPSEQ DS 0H
SPACE 1
OPEN (QSDCB,(INPUT)),MODE=24
SPACE 2
LH R15,QSDCBLN          Get the maximum Rec size
ST R15,MAXLENF          and save it
SPACE 1
TM QSDCBRF,DCBRECVD    What is the RECFM?
BNO OPSEQF              Fixed
B OPSEQV                Variable
SPACE 1
OPSEQF DS 0H
SPACE 1
MVI RECFM,C'F'          Set Fixed
B OPSEQX
SPACE 1
OPSEQV DS 0H

```

```

SPACE 1
MVI RECFM,C'V'          Set Variable
B   OPSEQX
SPACE 1
OPSEQX DS 0H
EJECT
SPACE 2
* =====
* =
* =          Now connect to the MQ Manager
* =
* =====
SPACE 2
* =====> <=====
* =====> The first thing to do is to make contact with <=====
* =====> the required Queue Manager by doing the MQCONN <=====
* =====> verb. <=====
* =====> <=====
* =====> If this fails, then processing cannot continue <=====
* =====> so a diagnostic is issued and processing stops <=====
* =====> with a Return Code of 16. <=====
* =====> <=====
SPACE 1
DOCONN DS 0H
SPACE 1
MVC THISCOMM,=CL40'MQ CONNECT'  Connect to QM
SPACE 1
CALL MQCONN, (REQQM,QMH,QRC,QAC)
SPACE 1
MVC LASTRC,QRC          Save result
MVC LASTAC,QAC
SPACE 1
CLC QRC,=AL4(MQCC_OK)    Connect Worked?
BE DOCONNX              Yes - Proceed
*                       No - state error
SPACE 1
LA R1,QRC               Convert RC
LA R2,QRCH
BAL R7,CONV4TO8
LA R1,QAC               Convert AC
LA R2,QACH
BAL R7,CONV4TO8
SPACE 1
MVC RC,=F'16'          Set Return Code
SPACE 1
MVC PLINE1,MSG2        Say Error
MVC PLINE1+017(48),REQQM on QM
MVC PLINE1+081(08),QRCH and RC/AC
MVC PLINE1+097(08),QACH

```

```

        BAL R7,PRINT          Print Error line
        B   EXIT             and Exit
        SPACE 2
DOCONNX DS 0H
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =           Open the requested queue
* =
* =====
        SPACE 2
* =====> <=====
* =====> Open the required Queue for output processing. <=====
* =====>
* =====> If this fails, then processing cannot continue <=====
* =====> so a diagnostic is issued and processing stops <=====
* =====> with a Return Code of 16. <=====
* =====> <=====
        SPACE 1
DOQOP  DS 0H
        SPACE 1
        MVC THISCOMM,=CL40'MQOPEN'          Open the Command Q
        MVC LQOD_OBJECTNAME,WANTQNAM       Set Queue Name
        SPACE 1
        CALL MQOPEN,(QMH,LQOD,LQOP,QH,QRC,QAC)
        SPACE 1
        MVC LASTRC,QRC                     Save result
        MVC LASTAC,QAC
        SPACE 1
        CLC QRC,=AL4(MQCC_OK)              Open Worked?
        BE  DOQOPX                          Yes - Proceed
*                                           No - state error
        SPACE 1
        LA  R1,QRC                          Convert RC
        LA  R2,QRCH
        BAL R7,CONV4TO8
        LA  R1,QAC                          Convert AC
        LA  R2,QACH
        BAL R7,CONV4TO8
        SPACE 1
        MVC RC,='F'16'                      Set Return Code
        SPACE 1
        MVC PLINE1,MSG3                     Say Error
        MVC PLINE1+018(48),WANTQNAM         Give the Queue Name
        MVC PLINE1+082(08),QRCH            Say Error RC/AC
        MVC PLINE1+098(08),QACH
        BAL R7,PRINT          Print Error line

```

```

        B      EXIT                                and Exit
        SPACE 1
DOQOPX  DS     0H
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =      Obtain the data length for the queue      =
* =
* =====
        SPACE 2
* =====>                                     <=====
* =====> In order to see if the Queue needs altering <=====
* =====> to accept the maximum length records in the <=====
* =====> file, do a MQINQ on it to get the maximum <=====
* =====> record length.                               <=====
* =====>
* =====> If this fails, then processing cannot continue <=====
* =====> so a diagnostic is issued and processing stops <=====
* =====> with a Return Code of 12.                   <=====
* =====>
        SPACE 1
DOQINQ  DS     0H
        SPACE 1
        MVC   THISCOMM,=CL40'MQINQ'                Open the Command Q
        SPACE 1
        CALL  MQINQ,(QMH,QH,QISC,QIS,QIIAC,QIIA,QICAL,QICA,QRC,QAC)
        SPACE 1
        MVC   LASTRC,QRC                          Save result
        MVC   LASTAC,QAC
        SPACE 1
        CLC   QRC,=AL4(MQCC_OK)                    Inquire Worked?
        BE    DOQINQ1                               Yes - Proceed
*****
* Slightly butchered by PRJ to allow qremote and aliasq
*****
        CLC   QAC,=AL4(2068)                        Remote Queue ?
        MVC   MAXLENQ,=F'32768'                    Default to 32k max msg
        BE    DOQINQ1                               Proceed with remote queue
*****
* End of butchery
*****
        SPACE 1
        LA    R1,QRC                                Convert RC
        LA    R2,QRCH
        BAL   R7,CONV4T08
        LA    R1,QAC                                Convert AC
        LA    R2,QACH

```



```

        BAL  R7,CONV4T08
        SPACE 1
        MVC  RC,=F'12'                Set Return Code
        SPACE 1
        MVC  PLINE1,MSG4                Say Error
        MVC  PLINE1+017(48),WANTQNAM    Give the Queue Name
        MVC  PLINE1+081(08),QRCH        Say Error RC/AC
        MVC  PLINE1+097(08),QACH
        BAL  R7,PRINT                    Print Error line
        B    EXIT                        and Exit
        SPACE 1
DOQINQ1 DS  0H
        SPACE 2
        MVC  MAXLENQ,QIIA                Save Max Q message len
        SPACE 1
DOQINQX DS  0H
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =      Stop if the queue is too small
* =
* =====
        SPACE 2
* =====> <=====
* =====> If the maximum message length for the queue <=====
* =====> is LESS than that for the file to be loaded, <=====
* =====> then processing stops with a Return Code of 4. <=====
* =====> <=====
* =====> <=====
        SPACE 1
DOQSML DS  0H
        SPACE 1
        CLC  MAXLENQ,MAXLENF            Queue Message too small?
        BNL  DOQSMLX                    No - leave q alone
*                                          Yes - stop
        SPACE 1
        MVC  RC,=F'4'                Set Return Code
        SPACE 1
        MVC  PLINE1,MSG5
        MVC  PLINE1+040(48),WANTQNAM    Set Queue Name
        LA   R1,MAXLENF                Convert File LRECL
        LA   R2,PLINE1+15
        BAL  R7,CONV4T06
        LA   R1,MAXLENQ                and Queue Max Msg Length
        LA   R2,PLINE1+108
        BAL  R7,CONV4T06
        BAL  R7,PRINT                    Print Error line

```

```

        B      EXIT                                and Exit
        SPACE 1
DOQSMLX DS    0H
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =      Q load loop - read the SYSIN FILE      =
* =
* =====
        SPACE 2
*      =====>                                <=====
*      =====> The SYSIN file is read to obtain a Queue name <=====
*      =====> to be deleted. Note that End-of-File etc.      <=====
*      =====> processing exits this loop.                    <=====
*      =====>
*      =====> Locate mode I/O is being used.                 <=====
*      =====>
        SPACE 1
CQLPO   DS    0H
        SPACE 1
        MVC   THISCOMM,=CL40'GET'
        SPACE 1
        GET   QSDCB
        SPACE 1
        CLI   RECFM,C'F'                                Test the RECFM
        BE    CQLPOF                                    Fixed
        B     CQLPOV                                    Variable
        SPACE 1
CQLPOF  DS    0H
        SPACE 1
        LH    R15,QSDCBLN                                Get record length
        ST    R15,CURECLEN                                and save it
        ST    R1,QSREC                                    Save the record address
        B     CQLPOU
        SPACE 1
CQLPOV  DS    0H
        SPACE 1
        LH    R15,0(,R1)                                Get record length
        SH    R15,=H'4'                                  remove LL00 prefix
        ST    R15,CURECLEN                                and save it
        LA    R1,4(,R1)                                  Point past LL00 prefix
        ST    R1,QSREC                                    and save data addr
        B     CQLPOU
        SPACE 1
CQLPOU  DS    0H
        SPACE 1
        L     R14,RECNO                                Bump up counter of

```

```

                LA    R14,1(,R14)                loaded records
                ST    R14,RECNO
                SPACE 2
CQLP0X         DS    0H
                SPACE 2
                EJECT
                SPACE 2
* =====
* =
* =      Q load loop - put the message into the Queue      =
* =
* =====
                SPACE 2
*
* =====>                                     <=====
*
* =====> The record is now sent to the queue with      <=====
*
* =====> a MQPUT verb. The operation is under          <=====
*
* =====> Syncpoint control.                            <=====
*
* =====>                                     <=====
*
* =====> If this fails, then processing cannot continue <=====
*
* =====> so a diagnostic is issued and processing stops <=====
*
* =====> with a Return Code of 8.                      <=====
*
* =====>                                     <=====
*
* =====> A failure from now on in the loop will lead to <=====
*
* =====> the CSQBCMT Syncpoint call NOT being done, so <=====
*
* =====> any dangling messages will be deleted by MQM. <=====
*
* =====>                                     <=====
                SPACE 1
CQLP1         DS    0H
                SPACE 1
                MVC   THISCOMM,=CL40'MQPUT'          Set command
                SPACE 1
                MVC   PL1RECP,QSREC                 Set record address
                SPACE 2
                LA    R1,PLPUT1                     Address MQPUT parmlist
                L     R15,=V(MQPUT)                 Get MQPUT address
                BALR  R14,R15                        Dp the MQPUT
                SPACE 1
                MVC   LASTRC,QRC                    Save result
                MVC   LASTAC,QAC
                SPACE 1
                CLC   QRC,=AL4(MQCC_OK)             Put Worked?
                BE    CQLP1X                          Yes - Proceed
*
*                                                    No - state error
                SPACE 1
                LA    R1,QRC                          Convert RC
                LA    R2,QRCH
                BAL   R7,CONV4TO8
                LA    R1,QAC                          Convert AC
                LA    R2,QACH

```

```

        BAL  R7,CONV4T08
        SPACE 1
        MVC  RC,='8'           Set Return Code
        SPACE 1
        MVC  PLINE1,MSG6       Say Error
        MVC  PLINE1+017(48),WANTQNAM  Say Failed defined Q
        MVC  PLINE1+081(08),QRCH   Say Error RC/AC
        MVC  PLINE1+097(08),QACH
        BAL  R7,PRINT          Print Error line
        B    EXIT              and Exit
        SPACE 1
CQLP1X DS  0H
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =      Q load loop - reloop
* =
* =====
        SPACE 2
* =====> <=====
* =====> Now that a record has made it to the queue, <=====
* =====> simply get hold of the next record by looping. <=====
* =====> (EOF on SYSUT1 exits the loop). <=====
* =====> <=====
        SPACE 1
CQLP3  DS  0H
        SPACE 1
        B    CQLP0             Read next file record
        SPACE 1
CQLP3X DS  0H
        SPACE 1
        SPACE 2
        EJECT
        SPACE 2
* =====
* =
* =      SYSUT1 processing - End of file
* =
* =====
        SPACE 2
* =====> <=====
* =====> This routine is entered via the DCB when the <=====
* =====> SYSUT1 file hits End-of-File. This causes the <=====
* =====> last message (of zero bytes) to be written <=====
* =====> to the queue. <=====
* =====> <=====
* =====> However, first close the file of loaded <=====

```

```

*          =====> records.                                     <=====
*          =====>                                           <=====
SPACE 2
QSEOF DS   0H
SPACE 1
CLOSE QSDCB
SPACE 1
MVC  THISCOMM,=CL40'MQPUT LAST'   Set command
SPACE 1
MVC  PLIRECP,=F'0'                 Set no record
MVC  CURECLEN,=F'0'                of zero bytes
L    R14,RECNO                     Bump up counter of
LA   R14,1(,R14)                   loaded records
ST   R14,RECNO
SPACE 2
LA   R1,PLPUT1                     Address MQPUT parmlist
L    R15,=V(MQPUT)                 Get MQPUT address
BALR R14,R15                       Dp the MQPUT
SPACE 1
MVC  LASTRC,QRC                    Save result
MVC  LASTAC,QAC
SPACE 1
CLC  QRC,=AL4(MQCC_OK)             Put Worked?
BE   QSEOFX                         Yes - Proceed
*                                     No - state error
SPACE 1
LA   R1,QRC                         Convert RC
LA   R2,QRCH
BAL  R7,CONV4T08
LA   R1,QAC                         Convert AC
LA   R2,QACH
BAL  R7,CONV4T08
SPACE 1
MVC  RC,=F'8'                      Set Return Code
SPACE 1
MVC  PLINE1,MSG7                   Say Error
MVC  PLINE1+022(48),WANTQNAM       Say Failed defined Q
MVC  PLINE1+086(08),QRCH          Say Error RC/AC
MVC  PLINE1+103(08),QACH
BAL  R7,PRINT                       Print Error line
B    EXIT                           and Exit
SPACE 1
QSEOFX DS   0H
SPACE 2
EJECT
SPACE 2
* =====
* =
* = End of Q loading - Syncpoint before closing the queue. =

```

```

* =
* =====
SPACE 2
* =====> <=====
* =====> All the requested records have been loaded. <=====
* =====>
* =====> Thus, Syncpoint the queue to ensure they <=====
* =====> have all made it. <=====
* =====>
* =====> If this fails, then processing cannot continue <=====
* =====> so a diagnostic is issued and processing stops <=====
* =====> with a Return Code of 12. <=====
* =====> <=====
SPACE 1
SYNC DS 0H
SPACE 1
MVC THISCOMM,=CL40'MQ SYNCPOINT'
SPACE 1
CALL CSQBCMT,(QMH,QRC,QAC)
SPACE 1
MVC LASTRC,QRC Save result of Syncpoint
MVC LASTAC,QAC
SPACE 1
CLC QRC,=AL4(MQCC_OK) Committ Worked?
BE SYNCX Yes - Proceed
* No - state error
SPACE 1
LA R1,QRC Convert RC
LA R2,QRCH
BAL R7,CONV4TO8
LA R1,QAC Convert AC
LA R2,QACH
BAL R7,CONV4TO8
SPACE 1
MVC RC,=F'12' Set Return Code
SPACE 1
MVC PLINE1,MSG8 Say Error
MVC PLINE1+029(48),WANTQNAM Say Failed defined Q
MVC PLINE1+092(08),QRCH Say Error RC/AC
MVC PLINE1+108(08),QACH
BAL R7,PRINT Print Error line
B EXIT and Exit
SPACE 1
SYNCX DS 0H
SPACE 2
EJECT
SPACE 2
* =====
* =

```

```

* =      End of Q loading - Close the Queue      =
* =
* =====
SPACE 2
* =====>                                     <=====
* =====> All the requested records have been loaded. <=====
* =====>                                     <=====
* =====> Thus, MQCLOSE the Queue with its Committed <=====
* =====> records.                               <=====
* =====>                                     <=====
* =====> After the Queue has been closed, then <=====
* =====> this program stops MQ usage by doing a MQDISC <=====
* =====> operation.                             <=====
* =====>                                     <=====
* =====> Prior errors do not go through this logic. <=====
* =====>                                     <=====
* =====> If the Commit is not done, then the added <=====
* =====> records will be backed out, so this program <=====
* =====> step can be rerun if required.          <=====
* =====>                                     <=====
SPACE 1
QC DS    0H
SPACE 1
MVC  THISCOMM,=CL40'MQ CLOSE'
CALL  MQCLOSE,(QMH,QH,LQCP,QRC,QAC)
SPACE 1
MVC  LASTRC,QRC           Save result of Close
MVC  LASTAC,QAC
SPACE 1
CLC  QRC,=AL4(MQCC_OK)    Close Worked?
BE   QC1                 Yes - Proceed
*                               No - state error

SPACE 1
LA   R1,QRC              Convert RC
LA   R2,QRCH
BAL  R7,CONV4TO8
LA   R1,QAC              Convert AC
LA   R2,QACH
BAL  R7,CONV4TO8
SPACE 1
MVC  RC,=F'16'           Set Return Code
SPACE 1
MVC  PLINE1,MSG9         Say Error
MVC  PLINE1+019(48),WANTQNAM Say Failed defined Q
MVC  PLINE1+083(08),QRCH Say Error RC/AC
MVC  PLINE1+099(08),QACH
BAL  R7,PRINT            Print Error line
B    EXIT                and Exit
SPACE 1

```

```

QC1    DS    OH
       SPACE 1
       MVC   PLINE1,MSG10           Say OK
       LA    R1,RECNO              Convert Number of
       LA    R2,PLINE1+01          records loaded
       BAL   R7,CONV4TO6
       MVC   PLINE1+044(48),WANTQNAM Denote the Queue
       BAL   R7,PRINT              Print succeeded line
       SPACE 1
       MVC   THISCOMM,=CL40'MQ DISC'
       CALL  MQDISC,(QMH,QRC,QAC)
       SPACE 1
       MVC   LASTRC,QRC            Save result of DISC
       MVC   LASTAC,QAC
       SPACE 1
       CLOSE SYSPRINT
       SPACE 1
       B     RETURN                and return to caller
       SPACE 1
QCX    DS    OH
       SPACE 2
       EJECT
       SPACE 2
* =====
* =
* =      Exit processing - error - Rollback any updates and end  =
* =
* =====
       SPACE 2
EXIT   DS    OH
       SPACE 2
*      =====> <=====
*      =====> The common (error) Exit point ensures that the <=====
*      =====> SYSPRINT file is closed (to prevent data loss),<=====
*      =====> all Queue Updates are lost, the Queue is <=====
*      =====> not used, and access to the Queue Manager <=====
*      =====> ended. <=====
*      =====> <=====
       SPACE 2
       CLOSE SYSPRINT
       SPACE 1
       MVC   THISCOMM,=CL40'MQ ROLLBACK'
       CALL  CSQBBAK,(QMH,QRC,QAC)
       SPACE 1
       MVC   LASTRC,QRC            Save result of Rollback
       MVC   LASTAC,QAC
       SPACE 1
       MVC   THISCOMM,=CL40'MQ CLOSE'
       CALL  MQCLOSE,(QMH,QH,LQCP,QRC,QAC)

```



```

SPACE 1
MVC  LASTRC,QRC           Save result of Close
MVC  LASTAC,QAC
SPACE 1
MVC  THISCOMM,=CL40'MQ DISC'
CALL  MQDISC,(QMH,QRC,QAC)
SPACE 1
MVC  LASTRC,QRC           Save result of DISC
MVC  LASTAC,QAC
SPACE 1
B     RETURN              and return to caller
SPACE 2
EXITX DS  0H
SPACE 2
EJECT
SPACE 2

* =====
* =
* =           Return to caller
* =
* =====

SPACE 2
*  =====> <=====
*  =====> Now return to the caller by restoring all <=====
*  =====> Registers and supplying the apt Return Code <=====
*  =====> in R15. <=====
*  =====> <=====

SPACE 1
RETURN DS  0H
SPACE 2
L     R15,RC              Get the Return Code
LM    R0,R14,SAVE1       Load Registers
SPACE 2
BSM   R14,R14            Return to caller
SPACE 2
EJECT
SPACE 2

* =====
* =           Convert XL4 -> CL8
* =
* =
* =           R1 -> Source 4 Hex bytes
* =           R2 -> Receiving 8 Character bytes
* =           R7 -> linkage
* =
* =====

SPACE 2
CONV4T08 DS  0H
SPACE 1
XR    R14,R14            Clear Work Reg

```

```

L      R14,0(,R1)          Load Source Bytes
CVD   R14,WORK4T08       Put in Packed format
UNPK  PACK4T08,WORK4T08  Put into Character format
OI    PACK4T08+15,X'F0'  Make Readable
MVC   0(8,R2),PACK4T08+8 Return Value
BR    R7                  Return to caller
SPACE 2
EJECT
SPACE 2
* =====
* =          Convert XL4 -> CL16          =
* =
* =          R1 -> Source 4 Hex bytes      =
* =          R2 -> Receiving 16 Character bytes =
* =          R7 -> linkage                  =
* =
* =====
SPACE 2
CONV4T06 DS 0H
SPACE 1
XR    R14,R14             Clear Work Reg
L     R14,0(,R1)          Load Source Bytes
CVD   R14,WORK4T06       Put in Packed format
UNPK  PACK4T06,WORK4T06  Put into Character format
OI    PACK4T06+15,X'F0'  Make Readable
MVC   0(16,R2),PACK4T06+0 Return Value
BR    R7                  Return to caller
SPACE 2
EJECT
SPACE 2
* =====
* =          Print a line to SYSPRINT from PLINE1 =
* =
* =          R7 -> linkage                  =
* =
* =====
SPACE 2
PRINT DS 0H
SPACE 1
PUT   SYSPRINT,PLINE
SPACE 1
BR    R7
* =====
* =
* =          Save Areas                    =
* =
* =====
SPACE 2
SAVE1 DC 16F'0'          Initial Save Area

```

```

SPACE 1
SAVEOS1 DC 18F'0' OS-type Save Area
SPACE 1
WSAVE DC 16F'0' WTO Save Area
SPACE 1
SPACE 2
EJECT
SPACE 2

* =====
* =
* = DCB Areas =
* =
* =====

SPACE 2
* =====> <=====
* =====> SYSIN contains the name of the queue to load <=====
* =====> <=====

SPACE 1
DS OF
SYSIN DCB DDNAME=SYSIN,
RECFM=F,
LRECL=80,
BLKSIZE=80,
DSORG=PS,
BUFNO=1,
MACRF=GM,
EODAD=SYSINE

SPACE 2
* =====> <=====
* =====> SYSPRINT is used to print info <=====
* =====> <=====

SPACE 1
DS OF
SYSPRINT DCB DDNAME=SYSPRINT,
RECFM=VBA,
LRECL=133,
BLKSIZE=137,
DSORG=PS,
BUFNO=1,
MACRF=PM

SPACE 2
* =====> <=====
* =====> QSDCB is the DCB used to access the SYSUT1 file <=====
* =====> <=====

SPACE 1
DS OF
QSDCB DCB DDNAME=SYSUT1,
DSORG=PS,
BUFNO=1,

```

```

                MACRF=GL,
                EODAD=QSEOF
SPACE 2
QSDCBX DS OXL1
        ORG QSDCB+(DCBLRECL-DCBRELAD)
QSDCBLN DS H                                Record length
        ORG QSDCB+(DCBRECFM-DCBRELAD)
QSDCBRF DS XL1                              Record format
        ORG QSDCBX
        EJECT
SPACE 2
* =====
* =
* =                Variables                =
* =
* =====
SPACE 2
PARMADDR DC F'0'                            Parm address
THISCOMM DC CL40' '                          Current Command
LASTRC   DC F'0'                            RC
LASTAC   DC F'0'                            AC
RC       DC F'0'                            Program Return Code
SPACE 2
REQQM    DC CL(MAXQML)' '                    Local QM name
WANTQNAM DC CL(MAXQML)' '                    Obtained Q name to delete
QMH      DC F'0'                            Queue Manager Handle
QH       DC F'0'                            Queue Handle
QRC      DC F'0'                            MQ Return Code
QAC      DC F'0'                            MQ Reason Code
QRCH     DC CL8' '                          MQ Return Code - Printable
QACH     DC CL8' '                          MQ Reason Code - Printable
SPACE 2
MAXLENF  DC F'0'                            Max length of File record
MAXLENQ  DC F'0'                            Queue
CURECLEN DC F'0'                            Current Len of file rec
RECNO    DC F'0'                            Number of records loaded
RECFM    DC CL1' '                          File format
SPACE 1
DS       0D
WORK4T08 DC XL8'00'                          General Converters
PACK4T08 DC CL16' '
WORK4T06 DC XL8'00'
PACK4T06 DC CL16' '
SPACE 1
QSREC    DC CL80' '                          SYSIN Input buffer
SPACE 1
PLINE    DC XL4'00850000'                    Print line - length
PLINE1   DC CL133' '                        - message
         DC CL10' '                        - overrun area

```

```

SPACE 2
EJECT
SPACE 2
* =====
* =
* =          Parm lists
* =
* =====

SPACE 2
* =====>                                     <=====
* =====> MQPUT Parm list                       <=====
* =====>                                     <=====

SPACE 1
DS    OD
PLPUT1 DS  OXL1
PL1QMH DC  AL4(QMH)          @QM Handle
PL1QH  DC  AL4(QH)           @Q Handle
PL1MD  DC  AL4(LQMD)         @Put Message descriptor
PL1PP  DC  AL4(LQPP)         @Put Message options
PL1MLEN DC AL4(CURECLEN)     @Put Message length
PL1RECP DC AL4(O)            @Message to write to Q
PL1RC  DC  AL4(QRC)          @Return Code
PL1AC  DC  AL4(QAC+X'80000000') @Reason Code

SPACE 2
EJECT
SPACE 2
* =====
* =
* =          Queue Definitions
* =
* =====

SPACE 2
* =====>                                     <=====
* =====> The Queue to load                       <=====
* =====>                                     <=====

SPACE 1
LQOD  CMQODA DSECT=NO,
      STRUCID=OD,
      VERSION=MQOD_VERSION_1,
      OBJECTTYPE=MQOT_Q,
      OBJECTNAME=,
      OBJECTMGRNAME=,
      DYNAMICQNAME=,
      ALTERNATEUSERID=,
      LIST=YES

SPACE 2
LQOP  DC  AL4(MQOO_OUTPUT+MQOO_INQUIRE+MQOO_SET+MQOO_FAIL_IF QUIES!
      CING) Open options
LQCP  DC  AL4(MQCO_NONE) Close options

```

```

SPACE 2
EJECT
SPACE 2
* =====> <=====
* =====> The MQPUT Message descriptor for the Queue. <=====
* =====>
* =====> Note that the MSGID, CORRELID and <=====
* =====> ACCOUNTINGTOKEN do NOT accept _NULL settings <=====
* =====> as implied by the APR. <=====
* =====> <=====
SPACE 1
LQMD CMQMDA DSECT=NO,
      STRUCID=MD,
      VERSION=MQMD_VERSION_1,
      REPORT=MQRO_NONE,
      MSGTYPE=MQMT_DATAGRAM,
      EXPIRY=MQEI_UNLIMITED,
      FEEDBACK=MQFB_NONE,
      FORMAT=MQFMT_NONE,
      PRIORITY=MQPRI_PRIORITY_AS_Q_DEF,
      PERSISTENCE=MQPER_NOT_PERSISTENT,
      MSGID=00,
      CORRELID=00,
      REPLYTOQ=,
      REPLYTOQMGR=,
      USERIDENTIFIER=,
      ACCOUNTINGTOKEN=00,
      APPLIDENTITYDATA=,
      PUTAPPLTYPE=MQAT_MVS,
      PUTAPPLNAME=,
      PUTDATE=,
      PUTTIME=,
      APPLORIGINDATA=,
      LIST=YES
SPACE 2
LQPP CMQPMOA DSECT=NO,
      STRUCID=PMO,
      VERSION=MQPMO_VERSION_1,
      OPTIONS=MQPMO_SYNCPOINT+MQPMO_NO_CONTEXT+MQPMO_FAIL_IF_Q
      UIESCING,
      CONTEXT=0,
      LIST=YES
SPACE 2
EJECT
SPACE 2
* =====>
* =
* = Inquire area
* =

```



```

RC 12345678 '
MSG8 DC CL(L'PLINE1)' COMMIT PROCESSING ON QUEUE 123456789012345|
678901234567890123456789012345678 FAILED WITH CC 1234567|
8 AND RC 12345678 '
MSG9 DC CL(L'PLINE1)' MQCLOSE FOR QUEUE 123456789012345678901234|
567890123456789012345678 FAILED WITH CC 12345678 AND RC |
12345678 '
MSG10 DC CL(L'PLINE1)' 1234567890123456 RECORDS LOADED INTO QUEUE|
123456789012345678901234567890123456789012345678 '
SPACE 2
LTOrg
99200000
SPACE 2
* =====
* =
* = End of processing =
* =
* =====
SPACE 2
EJECT
SPACE 2
* =====
* =
* = End of MQVPSTOQ program =
* =
* =====
SPACE 2
END VPSTOQ

```

Assemble and link JCL

```

//MQZG JOB (999,PRJ),'ITSO SA-A227-R',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M
//*JOBPARM L=999,SYSAFF=SC66
//*****
//* ASSEMBLE AND LINK A USER PROGRAM THAT ISSUES MQSERIES CALLS. *
//*****
//ASM EXEC PGM=ASMA90,PARM='DECK,NOOBJECT,LIST,XREF(SHORT)'
//SYSLIB DD DSN=MQ530.SCSQMACS,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&ASM,
// UNIT=SYSDA,DISP=(,PASS),
// SPACE=(400,(100,100,1))
//SYSPRINT DD SYSOUT=*
//MQZG JOB (999,PRJ),'ITSO SA-A227-R',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=0M

```



```

/*JOBPARM L=999,SYSAFF=SC66
//*****
/* ASSEMBLE AND LINK A USER PROGRAM THAT ISSUES MQSERIES CALLS.  *
//*****
//ASM EXEC PGM=ASMA90,PARM='DECK,NOBJECT,LIST,XREF(SHORT) '
//SYSLIB DD DSN=MQ530.SCSQMACS,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSPUNCH DD DSN=&&ASM,
// UNIT=SYSDA,DISP=(,PASS),
// SPACE=(400,(100,100,1))
//SYSPRINT DD SYSOUT=*
//SYSIN DD DSN=SDRES3.CNTL.ASM(MQVPSTOQ),DISP=SHR
//LKED EXEC PGM=IEWL,COND=(8,LE),
// PARM='SIZE=(900K,124K),RENT,LIST,AMODE=24,RMODE=24'
//SYSLMOD DD DSN=SDRES3.CNTL.LOADLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,DCB=BLKSIZE=1024,
// SPACE=(1024,(200,20))
//SYSPRINT DD SYSOUT=X
//CSQSTUB DD DSN=MQ530.SCSQLOAD,DISP=SHR
//SYSLIB DD DSN=MQ530.SCSQLOAD,DISP=SHR
//SYSLIN DD DSN=&&ASM,DISP=(OLD,DELETE)
// DD *
INCLUDE CSQSTUB(CSQBSTUB)
NAME MQVPSTOQ(R)
/*

```

Execution JCL

```

//MQZG JOB (999,PRJ),'ITSO SA-A227-R',CLASS=A,MSGCLASS=T,
// NOTIFY=&SYSUID,TIME=1440,REGION=OM
/*JOBPARM L=999,SYSAFF=SC62
//*****
/* FUNCTION: *
/* THIS JCL WILL POPULATE A Q WITH DATA CONTAINED IN A *
/* FILE SPECIFIED IN THE SYSUT1 DD CARD. THE QUEUE IS *
/* NAMED IN SYSIN *
/* *
/* DATE : OCT 15TH 2002 *
/* AUTHOR : PETER RHYS-JENKINS *
//*****
//LOAD EXEC PGM=MQVPSTOQ,PARM=(MQZ2)
//STEPLIB DD DSN=SDRES3.CNTL.LOADLIB,DISP=SHR
// DD DSN=MQ530.SCSQANLE,DISP=SHR
// DD DSN=MQ530.SCSQAUTH,DISP=SHR
// DD DSN=MQ530.SCSQLOAD,DISP=SHR
//SYSUT1 DD DSN=SDRES3.AS400.DATA,DISP=SHR

```

```
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
PORTFOLIO.BATCH.MQ400
/*
```

CICS trigger monitor program source

```
IDENTIFICATION DIVISION.
PROGRAM-ID. XXXXXXXX.
AUTHOR. XXXXXXXXXXXXXXXX
DATE-WRITTEN. MARCH 2000.
DATE-COMPILED.
*-----
* Copyright and Support
* =====
* This program is not supported in any way by IBM. It is
* distributed only to show techniques for MQ usage.
*
*
* Author: For ITS0,
* IBM Hursley Park,
* England.
*-----
* THIS PROGRAM WILL BE INITIATED BY AN MQTRIGGER.
* IT WILL BROWSE THE CALLING QUEUE AND START A TRANSACTION
* AND APPLY A USERID FOUND IN MQMD-USERID.
*
ENVIRONMENT DIVISION.
DATA DIVISION.
*-----
* WORKING STORAGE
*-----
WORKING-STORAGE SECTION.
01 HOLD-AREAS.
05 FILLER PIC X(36) VALUE
'*****BEGIN WORKING STORAGE HERE*****'.
05 WS-PROGRAM-ID PIC X(08) VALUE
'SMPP0005'.
05 WS-PARA PIC X(30) VALUE SPACES.
05 WS-ERROR-MSG.
07 WS-MSG-1 PIC X(22) VALUE SPACES.
07 WS-MSG-2 PIC X(20) VALUE SPACES.
07 FILLER PIC X(8) VALUE SPACES.

01 WS-PARM-LEN PIC S9(03) BINARY.
01 WS-PARM-DATA.
```

```

03 WS-PARM-TRANID          PIC X(04) VALUE SPACES.
03 FILLER                  PIC X(01) VALUE SPACES.
03 WS-PARM-Q-NAME         PIC X(48) VALUE SPACES.

01 WS-Q-NAME.
05 WS-Q-DEST              PIC X(04) VALUE 'MAS.'.
05 WS-Q-TRANID           PIC X(04) VALUE SPACES.
05 FILLER                 PIC X(40) VALUE SPACES.

01 MQ-QUE-RECORD.
02 FILLER                  PIC X(32000) VALUE SPACES.
*-----*
*   COPYBOOK CONTAINS INTERFACE FOR ERROR LOGGING
*-----*
*   COPY SMPCERRC.
*-----*
*           M Q U S E R F L
*   COPYBOOK -- LAYOUT FOR MQTM-USERS DATA TO BE USED BY THE
*           TRIGGER BRIDGE MODULE
*-----*
*   COPY MQUSERFL.
*-----*
*   MQ FIELDS
*-----*
01 MQ-SHARED-FIELDS.
03 MQ-OBJ-OPTS            PIC S9(9) BINARY.
03 MQ-OBJHAND             PIC S9(9) BINARY.
03 MQ-COMPCODE           PIC S9(9) BINARY.
03 MQ-REASON              PIC S9(9) BINARY.
03 MQ-BUFFLEN             PIC S9(9) BINARY VALUE +32000.
03 MQ-DATALEN            PIC S9(9) BINARY.

*-----*
*   MQ STRUCTURES
*-----*
01 MQM-TRIGGER-MESSAGE.
   COPY CMQTML.
01 MQ-OBJ-DESC.
   COPY CMQODV.

01 MQ-MSG-DESC.
   COPY CMQMDV.

01 MQ-GET-MSG-OPTS.
   COPY CMQGMV.

01 MQ-PUT-MSG-OPTS.
   COPY CMQPMV.
*-----*

```

```

**      MQ CONSTANTS
*-----*
01      MQ-CONSTANTS.
        COPY CMQV.
*****
***                VARIABLES                ***
*****

01      WS-VARIABLES.
        05 WS-CICS-RESPONSE-CODE    PIC S9(08) COMP.
        05 WS-CICS-RESPONSE-CODE2  PIC S9(08) COMP.
*****
***                WORKING STORAGE SWITCHES                ***
*****

01      WS-SWITCHES-AND-COUNTERS.
        03 WS-TRANS-ST              PIC S9(08) COMP.
        03 WS-USER-ID               PIC X(08) VALUE SPACES.
        03 WS-TRAN-ID               PIC X(4)  VALUE SPACES.
        03 WS-LITERALS.
            05 WS-LINK-MODULE        PIC X(8)  VALUE 'DB2STAT' .

        03 WS-EMPTY-QUEUE-SW        PIC X(01) VALUE SPACE.
            88 EMPTY-QUEUE           VALUE 'Y'.
            88 NOT-EMPTY-QUEUE       VALUE SPACE.
        03 WS-CONTROL-SW            PIC X(01) VALUE SPACE.
            88 CRITICAL-ERROR        VALUE 'Y'.
            88 CONTINUE-PROCESSING   VALUE SPACE.

01      WS-DB2-CHECK-COMMAREA.
        COPY DB2STCA.

01      FILLER                      PIC X(37) VALUE
        '***** WORKING STORAGE ENDS HERE *****'.

LINKAGE SECTION.

*****
PROCEDURE DIVISION.
*****

000000-MAIN-LOGIC.

        PERFORM 100000-SETUP-PROCESS THRU 100000-SETUP-EXIT.
        PERFORM 200000-OPEN-QUEUE   THRU 200000-OPEN-QUEUE-EXIT.

        IF CONTINUE-PROCESSING
            IF TRIGGER-CHK-DB2-SW = 'Y'
                PERFORM 680000-CHECK-DB2-RTN THRU 680000-CHECK-DB2-EXIT
            END-IF

```

```

        IF CONTINUE-PROCESSING
            PERFORM 300000-MQGET-RECORD THRU 300000-MQGET-EXIT
        IF CONTINUE-PROCESSING
            PERFORM 400000-START-PROCESS THRU
                400000-START-EXIT

            END-IF
            PERFORM 600000-CLOSE-QUEUE THRU 600000-CLOSE-EXIT
        END-IF
    END-IF.

    GO TO 999999-PROGRAM-EXIT.

000000-MAIN-LOGIC-EXIT.
EXIT.

*****
*           INITIALIZE AND SET-UP PROCESS           *
*****
100000-SETUP-PROCESS.

EXEC CICS RETRIEVE
        INTO(MQTM)
        RESP    (WS-CICS-RESPONSE-CODE)
        RESP2   (WS-CICS-RESPONSE-CODE2)
        NOHANDLE
END-EXEC.

IF WS-CICS-RESPONSE-CODE = DFHRESP(NORMAL)
    MOVE MQTM-QNAME          TO WS-Q-NAME
    MOVE MQTM-USERDATA      TO TRIGGER-USER-DATA-AREA
ELSE
    MOVE +53                TO WS-PARM-LEN
    EXEC CICS RECEIVE
        INTO( WS-PARM-DATA )
        LENGTH( WS-PARM-LEN )
        RESP (WS-CICS-RESPONSE-CODE)
        RESP2(WS-CICS-RESPONSE-CODE2)
        NOHANDLE
    END-EXEC

    IF WS-CICS-RESPONSE-CODE = DFHRESP(NORMAL) OR
       WS-CICS-RESPONSE-CODE = DFHRESP(EOC)
        IF WS-PARM-LEN < 6
            MOVE 'Y'          TO WS-CONTROL-SW
        ELSE
            INITIALIZE MQTM
            MOVE WS-PARM-Q-NAME TO WS-Q-NAME
                                MQTM-QNAME

```

```

        END-IF
        ELSE
        MOVE 'Y'                                TO WS-CONTROL-SW
        END-IF
    END-IF.

100000-SETUP-EXIT.
EXIT.

*****
*
*          OPEN INPUT QUEUE
*****
200000-OPEN-QUEUE.

*   LOAD THE MQ-OBJ-DESC STRUCTURE:
*-----
        MOVE WS-Q-NAME                        TO MQOD-OBJECTNAME.
        MOVE MQOT-Q                          TO MQOD-OBJECTTYPE.
*-----

*   LOAD THE MQ-OBJ-OPTS FIELD:
*-----
        COMPUTE MQ-OBJ-OPTS = MQOO-BROWSE      +
                                MQOO-SET      +
                                MQOO-FAIL-IF-QUIESCING.

        CALL 'MQOPEN' USING MQHC-DEF-HCONN,
                                MQ-OBJ-DESC,
                                MQ-OBJ-OPTS, MQ-OBJHAND,
                                MQ-COMPCODE, MQ-REASON.

        EVALUATE (MQ-COMPCODE)
            WHEN MQCC-OK
                CONTINUE
            WHEN OTHER
                MOVE 'Y'                                TO WS-CONTROL-SW
                IF MQ-REASON = MQRC-CONNECTION-QUIESCING
                    CONTINUE
                ELSE
                    MOVE 'MQOPEN FAILED QUEUE==>' TO WS-MSG-1
                    MOVE MQOD-OBJECTNAME TO WS-MSG-2
                    MOVE ERRQ-MQOPEN-I TO ERRQ-MQ-DB-COMMAND
                    PERFORM 900000-LOG-ERR
                END-IF
            END-EVALUATE.

200000-OPEN-QUEUE-EXIT.
EXIT.

```

```

*****
*           GET INPUT DATA FROM QUEUE           *
*****
300000-MQGET-RECORD.

      COMPUTE MQGMO-OPTIONS = MQGMO-BROWSE-FIRST  +
                               MQGMO-FAIL-IF-QUIESCING.
      MOVE MQMI-NONE           TO MQMD-MSGID.
      MOVE MQCI-NONE           TO MQMD-CORRELID.
      MOVE ZEROES              TO MQ-DATALEN.
      MOVE SPACES              TO MQ-QUE-RECORD.

      CALL 'MQGET'  USING MQHC-DEF-HCONN,
                       MQ-OBJHAND,
                       MQMD,
                       MQ-GET-MSG-OPTS,
                       MQ-BUFFLEN,
                       MQ-QUE-RECORD
                       MQ-DATALEN,
                       MQ-COMPCODE,
                       MQ-REASON.

      EVALUATE (MQ-COMPCODE)
      WHEN MQCC-OK
        CONTINUE
      WHEN OTHER
        MOVE 'Y'                TO WS-CONTROL-SW
        IF MQ-REASON = MQRC-NO-MSG-AVAILABLE OR
           MQRC-CONNECTION-QUIESCING
          CONTINUE
        ELSE
          MOVE 'MQGET FAILED ON INPUT ' TO WS-MSG-1
          MOVE MQOD-OBJECTNAME         TO WS-MSG-2
          MOVE MQ-OBJHAND              TO ERRQ-INPUT-MQ-OBJHAND
          MOVE ERRQ-MQGET-TRIG         TO ERRQ-MQ-DB-COMMAND
          PERFORM 900000-LOG-ERR
        END-IF
      END-EVALUATE.

300000-MQGET-EXIT.
EXIT.

*****
*           START QUE PROCESS                     *
*****
400000-START-PROCESS.

```

```

IF TRIGGER-TRANS-ID > SPACES
  MOVE TRIGGER-TRANS-ID          TO WS-TRAN-ID
ELSE
  MOVE 'NO MQ-USER-DATA FOUND-' TO WS-MSG-1
  MOVE 'MUST DEFINE USERDATA'  TO WS-MSG-2
  MOVE MQ-OBJHAND               TO ERRQ-INPUT-MQ-OBJHAND
  MOVE ERRQ-CICS-INQ            TO ERRQ-MQ-DB-COMMAND
  PERFORM 900000-LOG-ERR
  GO TO 400000-START-EXIT
END-IF.

```

```

EXEC CICS INQUIRE
  TRANSID (WS-TRAN-ID)
  STATUS  (WS-TRANS-ST)
  RESP    (WS-CICS-RESPONSE-CODE)
  RESP2   (WS-CICS-RESPONSE-CODE2)
END-EXEC.

```

```

IF WS-CICS-RESPONSE-CODE = DFHRESP(NORMAL)
  IF WS-TRANS-ST = 0023
    CONTINUE
  ELSE
    MOVE 'CICS INQ FOUND TRANID ' TO WS-MSG-1
    MOVE 'TO START IS DISABLED'  TO WS-MSG-2
    MOVE MQ-OBJHAND              TO ERRQ-INPUT-MQ-OBJHAND
    MOVE ERRQ-CICS-INQ          TO ERRQ-MQ-DB-COMMAND
    PERFORM 900000-LOG-ERR
    GO TO 400000-START-EXIT
  END-IF

```

```

ELSE
  MOVE '**CICS INQ FAILED ****' TO WS-MSG-1
  IF WS-CICS-RESPONSE-CODE = 0028
    MOVE 'TRANSID NOT DEFINED ' TO WS-MSG-2
  ELSE
    MOVE 'UNKNOWN ERR ON INQ '  TO WS-MSG-2
  END-IF
  MOVE MQ-OBJHAND              TO ERRQ-INPUT-MQ-OBJHAND
  MOVE ERRQ-CICS-INQ          TO ERRQ-MQ-DB-COMMAND
  PERFORM 900000-LOG-ERR
  GO TO 400000-START-EXIT
END-IF.

```

```

MOVE MQMD-USERIDENTIFIER      TO WS-USER-ID.

```

```

EXEC CICS START
  TRANSID (WS-TRAN-ID)
  FROM    (MQM-TRIGGER-MESSAGE)
  LENGTH  (LENGTH OF MQM-TRIGGER-MESSAGE)

```



```

        USERID (WS-USER-ID)
        RESP   (WS-CICS-RESPONSE-CODE)
        RESP2  (WS-CICS-RESPONSE-CODE2)
END-EXEC.

IF WS-CICS-RESPONSE-CODE = DFHRESP(NORMAL)
  CONTINUE
ELSE
  MOVE 'CICS START TRAN FAILED' TO WS-MSG-1
  MOVE 'TRANS NOT STARTED!!!'  TO WS-MSG-2
  MOVE MQ-OBJHAND                TO ERRQ-INPUT-MQ-OBJHAND
  MOVE ERRQ-CICS-START           TO ERRQ-MQ-DB-COMMAND
  PERFORM 900000-LOG-ERR
END-IF.

400000-START-EXIT.
EXIT.

*****
*           CLOSE ANY QUE PROCESS           *
*****
600000-CLOSE-QUEUE.

        MOVE MQCO-NONE                TO MQ-OBJ-OPTS.

        CALL 'MQCLOSE' USING MQHC-DEF-HCONN,
                               MQ-OBJHAND, MQ-OBJ-OPTS,
                               MQ-COMPCODE, MQ-REASON.

600000-CLOSE-EXIT.
EXIT.

*****
*           CHECK DB2 CONNECTIVITY CHECK   *
*****
680000-CHECK-DB2-RTN.

        MOVE SPACES                TO WS-DB2-CHECK-COMMAREA.

EXEC CICS LINK
  PROGRAM (WS-LINK-MODULE)
  COMMAREA (WS-DB2-CHECK-COMMAREA)
  LENGTH (LENGTH OF WS-DB2-CHECK-COMMAREA)
  RESP (WS-CICS-RESPONSE-CODE)

END-EXEC.

IF WS-CICS-RESPONSE-CODE = DFHRESP(NORMAL)
  CONTINUE

```

```

ELSE
  MOVE 'Y'                                TO WS-CONTROL-SW
  MOVE 'LINK OF DB2 CHECK MODU'          TO WS-MSG-1
  MOVE 'LE FAILED'                       TO WS-MSG-2
  MOVE MQ-OBJHAND                        TO ERRQ-INPUT-MQ-OBJHAND
  MOVE ERRQ-DB2-FAIL                     TO ERRQ-MQ-DB-COMMAND
  PERFORM 900000-LOG-ERR
  GO TO 680000-CHECK-DB2-EXIT
END-IF.

```

```

IF DB2-IS-DOWN
  MOVE 'Y'                                TO WS-CONTROL-SW
  MOVE 'CICS TO DB2 CONNECTIVI'         TO WS-MSG-1
  MOVE 'TY NOT ESTABLISHED'             TO WS-MSG-2
  MOVE MQ-OBJHAND                        TO ERRQ-INPUT-MQ-OBJHAND
  MOVE ERRQ-DB2-FAIL                     TO ERRQ-MQ-DB-COMMAND
  PERFORM 900000-LOG-ERR
END-IF.

```

```

680000-CHECK-DB2-EXIT.
EXIT.

```

```

*****
*                               LINK TO ERROR HANDLING PROGRAM                               *
*****

```

```

900000-LOG-ERR.

```

```

MOVE WS-PROGRAM-ID                      TO ERRQ-PROGRAM-ID.
MOVE WS-PARA                             TO ERRQ-PARAGRAPH.
MOVE WS-ERROR-MSG                        TO ERRQ-ERROR-MSG.
MOVE EIBERRCD                            TO ERRQ-EIBERRCD.
MOVE EIBDS                                TO ERRQ-EIBDS.
MOVE EIBFN                                TO ERRQ-EIBFN.
MOVE EIBRCODE                            TO ERRQ-EIBRCODE.
MOVE EIBTRNID                            TO ERRQ-EIBTRNID.
MOVE MQ-COMPCODE                         TO ERRQ-MQCC.
MOVE MQ-REASON                           TO ERRQ-MQRC.

```

```

EXEC CICS LINK
  PROGRAM ('XXXXXXXX')
  COMMAREA (ERRQ-COMM)
  LENGTH (LENGTH OF ERRQ-COMM)
  RESP (WS-CICS-RESPONSE-CODE)
  RESP2 (WS-CICS-RESPONSE-CODE2)
END-EXEC.

```

```

900000-LOG-ERR-EXIT.
EXIT.

```

```

*****

```

```

**                                PROGRAM EXIT ROUTINE                                **
*****
999999-PROGRAM-EXIT.
EXEC CICS RETURN END-EXEC.
GOBACK.

```

z/OS assembler echo program

```

ECHO      TITLE 'MQ SAMPLE GET-PRINT-PUT ECHO PROGRAM'
*****
* This program is a modified version of the sample program CSQ4BAA1 *
* distributed in the hlq.SCSQASMS dataset supplied with the base FMID *
* *
* The program has been modified to echo received messages back to the *
* ReplyToQ and ReplyToQMgr specified in the MQMD of the received *
* message. The returned message has a prefix of ECHO: placed in front *
* of the received message. The destination qmgr and qname are written *
* to the SYSOUT dataset along with the message data. *
* See the original CSQ4BAA1 for program documentation. *
* *
* For:      ITSO Redbook SA-A227 *
*           Hursley UK. *
*           Peter Rhys-Jenkins Candle Corp. *
* *
* JCL to Assemble and Link the program; *
* *
* //MQZG JOB (999,PRJ),'ITSO SA-A227-R',CLASS=A,MSGCLASS=T, *
* // NOTIFY=&SYSUID,TIME=1440,REGION=OM *
* /*JOBPARM L=999,SYSAFF=SC66 *
* //***** *
* /** ASSEMBLE AND LINK A USER PROGRAM THAT ISSUES MQSERIES CALLS. ** *
* //***** *
* //ASM EXEC PGM=ASMA90,PARM='DECK,NOOBJECT,LIST,XREF(SHORT)' *
* //SYSLIB DD DSN=MQ530.SCSQMACS,DISP=SHR *
* // DD DSN=SYS1.MACLIB,DISP=SHR *
* //SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(1,1)) *
* //SYSPUNCH DD DSN=&&ASM, *
* // UNIT=SYSDA,DISP=(,PASS), *
* // SPACE=(400,(100,100,1)) *
* //SYSPRINT DD SYSOUT=* *
* //SYSIN DD DSN=SDRES3.CNTL.ASM(MQECHO),DISP=SHR *
* //LKED EXEC PGM=IEWL,COND=(8,LE), *
* // PARM='SIZE=(900K,124K),RENT,LIST,AMODE=31,RMODE=24' *
* //SYSLMOD DD DSN=SDRES3.CNTL.LOADLIB,DISP=SHR *
* //SYSUT1 DD UNIT=SYSDA,DCB=BLKSIZE=1024, *
* // SPACE=(1024,(200,20)) *
* //SYSPRINT DD SYSOUT=X *

```

```

* //CSQSTUB DD DSN=MQ530.SCSQLOAD,DISP=SHR *
* //SYSLIB DD DSN=MQ530.SCSQLOAD,DISP=SHR *
* //SYSLIN DD DSN=&&ASM,DISP=(OLD,DELETE) *
* // DD * *
* INCLUDE CSQSTUB(CSQBSTUB) *
* NAME MQECHO(R) *
* /* *
* *
* *
* JCL to execute the program; *
* *
* //MQZGE JOB (999,PRJ),'ITSO SA-A227-R',CLASS=A,MSGCLASS=T, *
00010000 *
* // NOTIFY=&SYSUID,TIME=1440,REGION=OM *
00020000 *
* /*JOBPARM L=999,SYSAFF=SC62 *
00030000 *
* //***** *
* /** FUNCTION: **
* /** THIS JCL WILL BROWSE, PRINT and ECHO MESSAGES **
* /** NOTE: QMGR AND QNAME ARE IN PARM **
* /** **
* //***** *
* //LOAD EXEC PGM=MQECHO,PARM=('MQZ2,ATM.REQUEST.CC.CICS') *
* //STEPLIB DD DSN=SDRES3.CNTL.LOADLIB,DISP=SHR *
* // DD DSN=MQ530.SCSQANLE,DISP=SHR *
* // DD DSN=MQ530.SCSQAUTH,DISP=SHR *
* // DD DSN=MQ530.SCSQLOAD,DISP=SHR *
* //SYSPRINT DD SYSOUT=* *
* // *
* *
*****
EJECT
*****
* REGISTERS *
*****
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12

```

```

R13      EQU   13
R14      EQU   14
R15      EQU   15
*****
*  MQ API CONSTANTS                                     *
*****

          CMQA LIST=NO
*****
*  DSECTS USED BY THIS PROGRAM                         *
*****
WORKAREA DSECT
*
PARMLIST CALL ,(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
CALLLST CALL ,(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0),VL,MF=L
*
PRTREC   DS    0CL133                                Output data record
PRTCC    DS    CL1                                   Carriage control
PRTDATA  DS    CL132                                Data
SKIPLINE DS    CL1                                   Carriage control store
*
WORKDWRD DS    D                                     Used for data conversion
PAGENUM  DS    F                                     Page number counter
LINENUM  DS    F                                     Line number printed
MSGNUM   DS    F                                     Message count number
DATALENGTH DS F                                     Actual message length
OPTIONS  DS    F                                     Options
POPTIONS DS F                                       Options
COMPCODE DS F                                       Completion code
REASON   DS    F                                     Reason code
HCONN    DS    F                                     Connection handle
HOBJ     DS    F                                     Connection handle
OBJECT   DS    F                                     Object handle
EXITCODE DS    F                                     Exit return code
TIMEDEC  DS    CL16                                  Time
DATEDEC  DS    PL4                                   Date
DATECONV DS    0CL8                                  Used for data conversion
          ORG   DATECONV
          DS    CL1
DATE_YR  DS    CL4                                   The year
DATE_JN  DS    CL3                                   The Julian date
          ORG
PARMADDR DS    F                                     Address of parm field
PARMLEN  DS    H                                     Length of parm field
*
MQMNAME  DS    CL48                                  Queue manager name
MQMQUEUE DS    CL48                                  Queue name
*
MSGDATA  DS    CL132                                Used by print routine
DSPCOMP  DS    CL04                                  Display compcode

```

```

DSPREAS DS CL04 Display reaon
DSPMSGL DS CL08 Display message length
CONVAREA DS CL8 Used for data conversion
*
BUFFER DS CL80 Target for message
NEWBUFF DS CL80
WORKLEN EQU *-WORKAREA
EJECT
*****
* START OF PROGRAM *
*****
MQECHO CSECT
MQECHO AMODE 31
SAVE (14,12) Save register
LR R11,R15 Set program
LA R12,4095(R11) Set up second base
LA R12,1(R12)
USING MQECHO,R11,R12 Addressability
ST R13,SAVEAREA+4 Backward pointer
LR R2,R13 Save in R2
LA R13,SAVEAREA Set addr of out savearea
ST R13,8(,R2) Forward savearea
ST R1,PARMSAVE Save parmlist
GETMAIN RU,LV=WORKLEN,LOC=(RES,ANY) Get some storage
LR R10,R1 Address of storage
USING WORKAREA,R10 Set addressability
B MAIN Branch to MAIN process
*
EJECT
*****
* SECTION NAME : MAIN *
* *
* FUNCTION : Controls flow of program *
* *
* CALLED BY : MQECHO CSECT *
* *
* CALLS : MAININIT, MAINPARM, MAINCONN, MAINMSGs, MAINDISC *
* ENDPROG *
* *
*****
MAIN DS OH
BAL R6,MAININIT Initialize storage
BAL R6,MAINPARM Reads the parms
BAL R6,MAINCONN Connect to qmgr
BAL R6,MAINMSGs Gets messages from queue
BAL R6,MAINDISC Disconnect from qmgr
B ENDPROG Terminate program
*
EJECT

```

```

*****
* SECTION NAME : MAININIT *
* * *
* FUNCTION : Performs initialization *
* * *
* CALLED BY : MAIN *
* * *
* CALLS : PRINTHDR *
* * *
* RETURN : To Register 6 *
* * *
*****
MAININIT DS OH
          OPEN (SYSPRINT),MF=(E,PRINTOPN),MODE=31
*
          TIME DEC,TIMEDEC,ZONE=LT,DATEYTYPE=YYYYDDD,LINKAGE=SYSTEM
*
          ZAP DATEDEC,=P'0' Initialise
          MVO DATEDEC,TIMEDEC+8(4) Date in packed dec
          UNPK DATECONV,DATEDEC To unsigned zoned
          MVZ DATECONV+7(1),DATECONV+6 Decimal
          MVC DATEYR,DATE_YR Move date - year
          MVC DATEJN,DATE_JN Move date - Julian
*
          XR R0,R0 Zero register
          ST R0,PAGENUM Zero page number
          ST R0,MSGNUM Zero message number
          ST R0,LINENUM Zero line number
          ST R0,EXITCODE Default return code
          BAL R7,PRINTHDR Write print headings
          BR R6
*
          EJECT
*****
* SECTION NAME : MAINPARM *
* * *
* FUNCTION : Read the parameters passed to the program, and *
* incorrect number of parameters passed *
* * *
* CALLED BY : MAIN *
* * *
* CALLS : PRINTHDG, PRTLINE *
* * *
* RETURN : Normal to Register 6 *
* Error to ENDPROG *
* * *
*****
MAINPARM DS OH
          MVI MQMNAME,X'40' Space out first byte

```

```

MVC MQMNAME+1(L'MQMNAME-1),MQMNAME and initialize
MVI MQMQUEUE,X'40' Space out first byte
MVC MQMQUEUE+1(L'MQMQUEUE-1),MQMQUEUE and initialize
*
L R1,PARMSAVE Address of parm list
L R1,0(R1) Address of first parm
LH R5,0(R1) Length of parm
STH R5,PARMLEN Save away
LTR R5,R5 any data passed
BZ NOPARMS No names passed
*
TRANPDM DS OH
LA R3,2(R1) Advance to start of parm
LR R2,R5 Load length and reduce
BCTR R2,R0 for execute
EX R2,TRANSCAN Scan variable bytes
BC 4,TWOPARMS Comma imbedded in text
B NOPARM2 No .. issue error
*
TWOPARMS DS OH
ST R1,PARMADDR Address of the comma
CR R1,R3 Is comma first char
BNE PARM1MVE No then no default qmgr
BCTR R5,R0 Reduce length
MVI HDR2_QMN,C' ' Space out first byte
MVC HDR2_QMN+1(L'HDR2_QMN-1),HDR2_QMN and initialize
MVC MSGDATA,INF2 Move in the message
BAL R8,PRTLINE and print the line
L R0,LINENUM Current line number
LA R0,1(R0) Add one and
ST R0,LINENUM save again
LA R0,4 Set return code
ST R0,EXITCODE ready for exit
L R1,PARMADDR Load Addr of Comma
LA R4,MQMQUEUE addr of the queue
BCTR R5,R0 reduce length for execute
EX R5,ONLYPARAM parameter
B ENDPARMS exit from parms
*
PARM1MVE DS OH
SR R1,R3 Length of data
LA R4,MQMNAME Address for target
BCTR R1,R0 Reduce for execute
EX R1,MOVEPARAM Move the data
LA R3,2(R1,R3) past first parm + comma
LH R2,PARMLEN Original length
LA R1,2(R1) Add two to R1
SR R2,R1 Length of second parm
LR R1,R2 Load length for move

```



```

*
PARM2MVE DS   OH
          LA   R4,MQMQUEUE           Address of target
          BCTR R1,R0                 Reduce for execute
          EX   R1,MOVEPARM          Move second parm
          B    ENDPARMS             Exit from parms
*
NOPARM2   DS   OH
          MVC  MSGDATA,INF3         Move in the message
          BAL  R8,PRTLINE           and print the line
          LA   R0,4                 Set return code
          ST   R0,EXITCODE          ready for exit
          B    ENDPROG             End the program
*
NOPARMS   DS   OH
          MVC  MSGDATA,INF1         Move message to buffer
          BAL  R8,PRTLINE           Write the record
          LA   R0,4                 Set return code
          ST   R0,EXITCODE          ready for exit
          B    ENDPROG             End the program
*
ENDPARMS  DS   OH
          BAL  R7,PRINTHDG         Print rest of headings
          BR   R6                   Return to caller
*
          EJECT
*****
* SECTION NAME : MAINCONN          *
*                                                    *
* FUNCTION      : Connect to the queue manager *
*                                                    *
* CALLED BY    : MAIN              *
*                                                    *
* CALLS        : ERRCODE           *
*                                                    *
* RETURN       : Normal to Register 6 *
*              Error  to ENDPROG    *
*                                                    *
*****
MAINCONN   DS   OH
          XC   HCONN,HCONN         Null connection handle
*
          CALL MQCONN,              X
              (MQMNAME,            X
              HCONN,               X
              COMPCODE,            X
              REASON),             X
              MF=(E,PARMLIST),VL
*

```

```

        LA  R0,MQCC_OK           Expected compcode
        C   R0,COMP CODE        As expected ?
        BER R6                   Yes .. return to caller
*
        MVC INF4_TYP,=CL10'CONNECT '
        BAL R7,ERRCODE          Translate error
        LA  R0,8                 Set exit code
        ST  R0,EXITCODE         to 8
        B   ENDPROG             End the program
*
        EJECT
*****
* SECTION NAME : MAINMSGs *
*
* FUNCTION      : Read the messages from the queue *
*
* CALLED BY     : MAIN *
*
* CALLS         : ERRCODE, PRTLINE *
*
* RETURN        : to Register 6 *
*
*****
MAINMSGs DS  OH
          LA  R0,MQOT_Q           Object is a queue
          ST  R0,GOBJDESC_OBJECTTYPE In object type field
          MVC GOBJDESC_OBJECTNAME,MQMQUEUE Move queue name
          LA  R0,MQOO_INPUT_SHARED Indicate open is
          ST  R0,OPTIONS          Shared
*
          CALL MQOPEN,           X
              (HCONN,           X
               GOBJDESC,       X
               OPTIONS,        X
               HOBJ,           X
               COMP CODE,      X
               REASON),        X
              MF=(E,PARMLIST),VL
*
          LA  R0,MQCC_OK           Expected compcode
          C   R0,COMP CODE        As expected?
          BE  MSGSINIT            Yes .. continue
          MVC INF4_TYP,=CL10'OPEN '
          BAL R7,ERRCODE          Translate error
          LA  R0,8                 Set exit code
          ST  R0,EXITCODE         to 8
          BR  R6                   Return to disconnect from
*                                     qmgr and terminate program
*

```

```

MSGSSINIT DS  OH
            L   RO,GOPTS
            ST  RO,GETMSGOPTS_OPTIONS      Indicate get options
            L   RO,=F'6000000'           100 minutes 1
            ST  RO,GETMSGOPTS_WAITINTERVAL

*
*-----*
* Code segment which gets the message from the requested queue. *
*-----*
MSGSGETS DS  OH
            XC  MSGGDESC_CORRELID,MSGGDESC_CORRELID Null correlation id
            XC  MSGGDESC_MSGID,MSGGDESC_MSGID      Null message id
            MVC BUFFER,MSGCLEAR                   Blank buffer

*
            CALL MQGET,                          X
                (HCONN,                          X
                 HOBJ,                            X
                 MSGGDESC,                       X
                 GETMSGOPTS,                     X
                 BUFFERLENGTH,                   X
                 BUFFER,                         X
                 DATALENGTH,                   X
                 COMPCODE,                      X
                 REASON),                       X
                MF=(E,PARMLIST),VL

*
            LA  RO,MQCC_OK                       Load compcode MQCC_OK
            C   RO,COMPCODE                       As expected?
            BE  MSGSPRNT                          Yes .. print message

*
            LA  RO,MQCC_WARNING                  Load compcode MQCC_WARNING
            C   RO,COMPCODE                       As expected
            BE  MSGSPRNT                          Yes .. print message

*
            LA  RO,MQRC_NO_MSG_AVAILABLE        No more message?
            C   RO,REASON                        Yes .. then close
            BE  MSGSCLOS                          Otherwise must be
            B   MSGSERR                          an error

*
MSGSPRNT DS  OH
            LA  RO,MQRC_TRUNCATED_MSG_ACCEPTED Was the warning because
            C   RO,REASON                        message was too long?
            BE  GETMSGS                          Yes .. get the message
            LA  RO,MQRC_NONE                    Was it ok?
            C   RO,REASON
            BE  GETMSGS                          Yes .. get the message
            B   MSGSERR                          Its unacceptable

*
*-----*

```

```

* Code segment which prints the message *
*-----*
GETMSG DS  OH
      L   R1,MSGNUM          Load current msg number
      LA  R1,1(R1)          Increment it by 1
      ST  R1,MSGNUM          Save back for next time
*
      CVD R1,WORKDWRD        To packed decimal
      UNPK CONVAREA,WORKDWRD+4(4) Convert to zoned decimal
      MVZ CONVAREA+7(1),CONVAREA+6 Make it displayable
      MVC  REP1_MGN,CONVAREA  Move to display area
*
      L   R5,DATALENGTH      Load length of message
      CVD R5,WORKDWRD        To packed decimal
      UNPK CONVAREA,WORKDWRD+4(4) Convert to zoned decimal
      MVZ CONVAREA+7(1),CONVAREA+6 Make it displayable
      MVC  REP1_MGL,CONVAREA  Move to display area
*
      LA  R3,BUFFER          Addr of source data
      LA  R4,REP1_MSG        Addr of target area
      C   R5,BUFFERLENGTH    Message greater than 80
      BNH WRITELIN           Ok to write it
      L   R5,BUFFERLENGTH    Reset to maximum
*
WRITELIN DS  OH
      MVC  REP1_MSG,MSGCLEAR  Clear the message
      C   R5,ZEROREC         Is message zero length
      BE  WRITEREP           Write blank line
      BCTR R5,R0              Reduce by 1 for execute
      EX  R5,MOVEDATA        Move data into output rec
      B   WRITEREP           Write the message
*
WRITEREP DS  OH
      MVC  MSGDATA,REP1       Move in the message
      BAL  R8,PRTLINE         Go and print line
      L   R1,LINENUM          Current line number
      LA  R1,1(R1)           Add one
      ST  R1,LINENUM          Save new value
*-----*
* Code segment that echo's the message back to the reply to queue *
*-----*
      MVC  POBJDESC_OBJECTNAME,GMSGDESC_REPLYTOQ
      MVC  POBJDESC_OBJECTQMGRNAME,GMSGDESC_REPLYTOQMGR
      L   R0,POPTS            Get put options
      ST  R0,PUTMSGOPTS_OPTIONS Indicate get options
      MVC  NEWBUFF(5),=C'ECHO:' Insert eyecatcher
      MVC  NEWBUFF+5(75),BUFFER Insert original message
*
      CALL MQPUT1,

```

X

```

                                (HCONN,                                X
                                POBJDESC,                               X
                                PMSGDESC,                               X
                                PUTMSGOPTS,                            X
                                NEWBUFL,                               X
                                NEWBUFF,                               X
                                COMPCODE,                               X
                                REASON),                               X
                                MF=(E,CALLLST),VL
*
LA    RO,MQCC_OK                Expected compcode
C     RO,COMPCODE               As expected?
BE    PUT10K                    Yes .. continue
MVC   INF5_QNM,POBJDESC_OBJECTNAME Get q name
MVC   INF5_QMG,POBJDESC_OBJECTQMGRNAME qmgr name
L     R1,LINENUM                Current line number
LA    R1,1(R1)                 Add one to line number
ST    R1,LINENUM                Save again
MVC   MSGDATA,INF5             Move in the message
BAL   R8,PRTLINE               Print the line
MVC   INF4_TYP,=CL10'MQPUT1    '
BAL   R7,ERRCODE               Translate error
* Never LA    RO,8                Set exit code
* Give  ST    RO,EXITCODE        to 8
* In !  BR    R6                 Return to disconnect from
*
PUT10K EQU *                    Put worked
MVC   INF5_QNM,POBJDESC_OBJECTNAME Get q name
MVC   INF5_QMG,POBJDESC_OBJECTQMGRNAME qmgr name
L     R1,LINENUM                Current line number
LA    R1,1(R1)                 Add one to line number
ST    R1,LINENUM                Save again
MVC   MSGDATA,INF5             Move in the message
BAL   R8,PRTLINE               Print the line
B     MSGSGETS                  Continue
*-----*
* Code segment closes the queue *
*-----*
MSGSCLOS DS    OH
          LA    RO,MQCO_NONE      Indicate normal close
          ST    RO,OPTIONS        of the queue
*
CALL MQCLOSE,                                X
      (HCONN,                            X
      HOBJ,                                X
      OPTIONS,                             X
      COMPCODE,                             X
      REASON),                             X
      MF=(E,PARMLIST),VL

```

```

*
      LA  R0,MQCC_OK           Expected compcode
      C   R0,COMPCODE         As expected?
      BER R6                   Yes .. continue
      MVC INF4_TYP,=CL10'CLOSE '
      BAL R7,ERRCODE          Translate error
      LA  R0,8                 Set exit code
      ST  R0,EXITCODE         To 8
      BR  R6                   Return to caller
*
MSGGSERR DS  0H
          MVC INF4_TYP,=CL10'GET '
          BAL R7,ERRCODE          Translate error
          LA  R0,8                 Set exit code
          ST  R0,EXITCODE         To 8
          BR  R6                   Return to caller
*
          EJECT
*****
* SECTION NAME : MAINDISC *
* * * * *
* FUNCTION      : Disconnect from the queue manager *
* * * * *
* CALLED BY    : MAIN *
* * * * *
* CALLS        : ERRCODE *
* * * * *
* RETURN       : to Register 6 *
* * * * *
*****
MAINDISC DS  0H
          CALL MQDISC,          X
              (HCONN,          X
               COMPCODE,       X
               REASON),        X
              MF=(E,PARMLIST),VL
*
      LA  R0,MQCC_OK           Expected compcode
      C   R0,COMPCODE         As expected?
      BER R6                   Yes .. continue
      MVC INF4_TYP,=CL10'DISCONNECT'
      BAL R7,ERRCODE          Translate error
      LA  R0,8                 Set exit code
      ST  R0,EXITCODE         To 8
      BR  R6                   Return to caller
*
          EJECT
*****
* SECTION NAME : ENDPROG *

```

```

*
* FUNCTION      : Program termination
*
* CALLED BY    : MAIN
*
* CALLS        : PRTLINE
*
* RETURN       : leaves program
*
*****
ENDPROG DS    OH
        MVC   SKIPLINE,TWOLINES      Skip a line
        MVC   MSGDATA,INFO           End report message
        BAL   R8,PRTLINE             Go and print line
        CLOSE (SYSPRINT)            Close output file
        L     R15,EXITCODE           Load termination code
        L     R13,4(R13)             Readdr caller SAVEAREA
        ST    R15,16(R13)           Place in return code
        LM    R14,R12,12(R13)       Restore registers
        L     R14,12(,R13)          Load return address
        BR    R14                   Return to caller
*
        EJECT
*****
* SUBROUTINES
*****
*
        SPACE 4
*****
* SECTION NAME : PUTREC
*
* FUNCTION      : Print a line
*
* CALLED BY    : PRINTHDR,
*
* CALLS        : nothing
*
* RETURN       : to Register 9
*
*****
PUTREC DS    OH
        LA    R4,THEPUT             Address of put
        BSM   0,R4                 Set correct mode
THEPUT PUT    SYSPRINT,PRTREC      Get buffer address
        BSM   0,R9                 Set mode of return
*
        EJECT
*****
* SECTION NAME : PRINTHDR
*

```

```

*
* FUNCTION      : Print first header line
*
* CALLED BY    : MAININIT, PRTLINE
*
* CALLS        : PUTREC
*
* RETURN       : to Register 7
*
*****
PRINTHDR DS   OH
           LA   R1,1                Load 1 into register
           ST   R1,LINENUM          Reset the line number
           L    R1,PAGENUM          Get current page number
           LA   R1,1(R1)            Increment by 1
           ST   R1,PAGENUM          Save again
*
           CVD  R1,WORKDWRD         To packed decimal
           UNPK CONVAREA,WORKDWRD+4(4) Convert to zoned decimal
           MVZ  CONVAREA+7(1),CONVAREA+6 Make it displayable
           MVC  HDR1_PGE,CONVAREA+4 Move to display area
*
           MVC  HDR1_DTE,REPDTE     Move in the date
           MVC  PRTDATA,HDR1        Move record to buffer
           MVC  PRTCC,NEWPAGE       Carriage control to page
           BAL  R9,PUTREC           Write the record
           BR   R7
*
           EJECT
*****
* SECTION NAME : PRTLINE
*
* FUNCTION      : Print the next line
*
* CALLED BY    : MAINPARM, MAINMSG, ENDPROG, ERRCODE
*
* CALLS        : PRINTHDR, PRINTHDG, PUTREC
*
* RETURN       : to Register 8
*
*****
PRTLINE DS   OH
          L    R1,LINENUM          Current line number
          C    R1,MAXLINES         Reached max lines
          BNH  PRTLINE1           No need for new page
          BAL  R7,PRINTHDR        Print first heading
          BAL  R7,PRINTHDG        Print other headings
*
PRTLINE1 DS   OH

```



```

*
MVC  PRTDATA,HDR6           Second heading line
MVC  PRTCC,TWOLINES        After skip 1 line
BAL  R9,PUTREC             Go and print line
L    R1,LINENUM            Current line number
LA   R1,2(R1)              Add two to line number
ST   R1,LINENUM            Save again
*
MVI  PRTDATA,C' '          Blank line
MVC  PRTDATA+1(L'PRTDATA-1),PRTDATA
MVC  PRTCC,ONELINE        After skip 1 line
BAL  R9,PUTREC             Go and print line
L    R1,LINENUM            Current line number
LA   R1,1(R1)              Add one to line number
ST   R1,LINENUM            Save again
BR   R7                    Return to caller
*
EJECT
*****
* SECTION NAME : ERRCODE *
* * * * *
* FUNCTION      : Print an error message including compcode and *
*                reason *
* * * * *
* CALLED BY    : MAINCONN, MAINMSG, MAINDISC *
* * * * *
* CALLS        : PRTLINE *
* * * * *
* RETURN       : to Register 7 *
* * * * *
*****
ERRCODE DS OH
L      RO,COMP CODE        Translate compcode
CVD    RO,WORKDWRD        To packed decimal
UNPK   CONVAREA,WORKDWRD+4(4) Convert to zoned decimal
MVZ    CONVAREA+7(1),CONVAREA+6 Make it displayable
MVC    INF4_CC,CONVAREA+4 Move to display area
*
L      RO,REASON           Translate reason
CVD    RO,WORKDWRD        To packed decimal
UNPK   CONVAREA,WORKDWRD+4(4) Convert to zoned decimal
MVZ    CONVAREA+7(1),CONVAREA+6 Make it displayable
MVC    INF4_RC,CONVAREA+4 Move to display area
*
L      R1,LINENUM          Current line number
LA     R1,1(R1)            Add one to line number
ST     R1,LINENUM          Save again
MVC    MSGDATA,INF4        Move in the message
BAL    R8,PRTLINE          Print the line

```

```

BR      R7                      Return to caller
*
EJECT
*****
*  CONSTANTS, DATA STRUCTURES  *
*****
SAVEAREA DS    9D                Save area
PARMSAVE DS    F                  Save area for parms
NEWPAGE  DC    C'1'              New page indicator
ONELINE  DC    C' '               Write next line
TWO LINES DC    C'-'             Write after advance 1
BUFFERLENGTH DC F'80'           Buffer size
NEWBUFL  DC    F'80'             Buffer size
MSGCLEAR DC    CL80' '           Blank out message
MAXLINES DC    F'60'             Max lines per page
ZEROREC  DC    F'0'              Check for zero length
*
DS      OF
GOBJDESC CMQODA DSECT=NO,LIST=NO Object descriptor
POBJDESC CMQODA DSECT=NO,LIST=NO Object descriptor
GMSGDESC CMQMDA DSECT=NO,LIST=NO Get message descriptor
PMSGDESC CMQMDA DSECT=NO,LIST=NO Put message descriptor
GETMSGOPTS CMQGMOA DSECT=NO,LIST=NO Get message options
PUTMSGOPTS CMQPMOA DSECT=NO,LIST=NO Put message options
*
SPACE 4
*****
* EXECUTES *
*****
ONLYPARM MVC  O(*-*,R4),1(R1)
MOVEPARM MVC  O(*-*,R4),0(R3)
MOVEDATA MVC  O(*-*,R4),0(R3)
TRANSCAN TRT  O(*-*,R3),TRANTAB
*
SPACE 4
*****
* TRANSLATE TABLE TO SCAN FOR DELIMETER. *
*****
TRANTAB DS    0CL256
ORG     TRANTAB
DC     X'00000000000000000000000000000000' 00 - 0F
DC     X'00000000000000000000000000000000' 10 - 1F
DC     X'00000000000000000000000000000000' 20 - 2F
DC     X'00000000000000000000000000000000' 30 - 3F
DC     X'00000000000000000000000000000000' 40 - 4F
DC     X'00000000000000000000000000000000' 50 - 5F
DC     X'0000000000000000000000000400000000' 60 - 6F
DC     X'00000000000000000000000000000000' 70 - 7F
DC     X'00000000000000000000000000000000' 80 - 8F

```

```

DC X'00000000000000000000000000000000' 90 - 9F
DC X'00000000000000000000000000000000' A0 - AF
DC X'00000000000000000000000000000000' B0 - BF
DC X'00000000000000000000000000000000' C0 - CF
DC X'00000000000000000000000000000000' D0 - DF
DC X'00000000000000000000000000000000' E0 - EF
DC X'00000000000000000000000000000000' F0 - FF
ORG

```

*

EJECT

* FIELDS USED BY PRINT ROUTINE *

```

REPDTE DS OCL10
DATEYR DS CL04
DATEF1 DC CL01'/'
DATEJN DS CL03
DATEF2 DC CL02' '
*
HDR1 DS OCL132 First heading line
DC CL10' '
HDR1_DTE DS CL010
DC CL36' '
DC CL19'SAMPLE QUEUE REPORT'
DC CL38' '
DC CL05'PAGE '
HDR1_PGE DS CL04
DC CL10' '
*
HDR2 DS OCL132 Second heading line
DC CL25' '
DC CL29'MESSAGE QUEUE MANAGER NAME : '
HDR2_QMN DS CL48
DC CL30' '
*
HDR3 DS OCL132 Third heading line
DC CL41' '
DC CL13'QUEUE NAME : '
HDR3_QUE DS CL48
DC CL30' '
*
HDR4 DS OCL132 Fourth heading line
DC CL12' '
DC CL08'RELATIVE'
DC CL112' '
*
HDR5 DS OCL132 Fifth heading line
DC CL12' '
DC CL07'MESSAGE'

```

```

          DC    CL113' '
*
HDR6     DS    OCL132                Sixth heading line
          DC    CL11' '
          DC    CL18' NUMBER   LENGTH '
          DC    CL33'-----'
          DC    CL14' MESSAGE DATA '
          DC    CL33'-----'
          DC    CL23' '
*
REP1     DS    OCL132                First report line
          DC    CL11' '
REP1_MGN DS    CL08
          DC    CL01' '
REP1_MGL DS    CL08
          DC    CL01' '
REP1_MSG DS    CL80
          DC    CL20' '
*
INFO     DS    OCL132
          DC    CL48' '
          DC    CL35'***** END OF REPORT *****'
          DC    CL49' '
*
INF1     DS    OCL132
          DC    CL10' '
          DC    CL46'***** NO DATA PASSED TO PROGRAM. PROGRAM '
          DC    CL46'REQUIRES A QUEUE MANAGER NAME AND A QUEUE NAME '
          DC    CL30'. ***** '
*
INF2     DS    OCL132
          DC    CL25' '
          DC    CL46'***** NO QUEUE MANAGER NAME PASSED TO PRO'
          DC    CL25'GRAM - DEFAULT USED *****'
          DC    CL36' '
*
INF3     DS    OCL132
          DC    CL38' '
          DC    CL10'*****'
          DC    CL34' NO QUEUE NAME PASSED TO PROGRAM. '
          DC    CL10'*****'
          DC    CL40' '
*
INF4     DS    OCL132
          DC    CL13' '
          DC    CL32'***** AN ERROR OCCURRED IN '
INF4_TYP DS    CL10
          DC    CL20'. COMPLETION CODE = '
INF4_CC  DS    CL04

```

```

          DC   CL16' REASON CODE = '
INF4_RC  DS   CL04
          DC   CL33' *****
*
INF5     DS   0CL132
          DC   CL11' '
          DC   CL17'*** QUEUE NAME : '
INF5_QNM DS   CL48
          DC   CL08' QMGR : '
INF5_QMG DS   CL48
*
GOPTS   DC   AL4(MQGMO_ACCEPT_TRUNCATED_MSG+MQGMO_CONVERT+MQGMO_WAIT)
POPTS   DC   AL4(MQPMO_FAIL_IF QUIESCING)
        EJECT
*****
* DCBS ETC
*****
        PRINT NOGEN
SYSPRINT DCB  DDNAME=SYSPRINT,
              DSORG=PS,,
              LRECL=133,
              BLKSIZE=7980,
              MACRF=PM,
              RECFM=FBA
*
PRINTOPN OPEN (SYSPRINT,(OUTPUT)),MODE=31,MF=L
*
        PRINT GEN
        LTORG
*
*****
        END   MQECHO

```

'C' message exit to restrict queue access

```

/*****
/*
/*   Module Name : QCLAMP
/*
/*
/*   Define File:
/*
/*   The following constitutes the define file for linking this dll */
/*
LIBRARY      QCLAMP
DESCRIPTION 'MQ Exit DLL'

```

```

HEAPSIZE  8192
STACKSIZE 8192
EXPORTS  QCLAMP      @1
/*
/*      Copyright and Support
/*      =====
/*      This program is not supported in any way by IBM. It is
/*      distributed only to show techniques for MQ usage. It may
/*      be freely modified and adapted by your installation.
/*
/*
/*
/* For ITSO RedBook SAA227
/* Author: Marc Verhiel Candle Corp. marc_verhiel@candle.com
/*
/* Exit Purpose: This exit is intended to prevent unauthorized msgs
/* from reaching their intended destination.
/*
/* Setup. 1) set the inbound channel msgexit(qclamp.dll(QCLAMP))
/*        2) set the inbound channel msgdata(name_list)
/*        3) define the name_list specified in step (2) and add the queues
/*           you want don't want messages diverted for added to it.
/* The queues not in the namelist will be clamped down, hence the name of
/* this exit.
/*
/*      As a result the exit will:
/*
/*      divert messages destined to a queue that is the same as the
/*      queue except that it has a .SECURE added to it. For example
/*      if the queue nm is SYSTEM.COMMAND.INPUT and this queue is not
/*      in the namelist then the inbound
/*      msg is destined for that queue it will be diverted to
/*      SYSTEM.COMMAND.INPUT.SECURE
/*      4) The message on the .SECURE queue will then be processed by
/*      some application that will know how to deal with it. Example:
/*      the message has an MQSecure trailer and hence it would be
/*      decrypted and if the decrypt is succesful the message would
/*      be moved to the originally intended queue.
/*
/* Limitations: queues cannot be longer than 41 characters since we're
adding
/*              7 characters to them.
/*
/* Note: this exit is for demonstration purposes only. If you really intend
/* to use this exit in production you'll want to eliminate the printf's
/* as well as look at a better way than using namelists to get the queue
/* names. Preload queues by another process in shared memory perhaps?
/*****/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>

#ifdef WIN32
#include <unistd.h>
#endif

#ifdef WIN32
#include <ctype.h>
#include <windows.h>
#include <winbase.h>
#endif

#include <time.h>
#include <sys/timeb.h>

/* includes for MQI */
#include <cmqc.h>
#include <cmqxc.h>

FILE *fp;

void print_timestamp(void);
void dump_message(PMQBYTE pBuffer, PMQLONG pDataLength);

/*****
/* Exit Structure Definitions */
*****/

void MQStart() {};
void MQENTRY QCLAMP(
    PMQCXP    pMQCXP,
    PMQCD     pMQCD,
    PMQLONG   pDataLength,
    PMQLONG   pAgentBufferLength,
    PMQBYTE   pAgentBuffer,
    PMQLONG   pExitBufferLength,
    PMQPTR    pExitBufferAddr
)
{
    PMQXQH    pMQXQH;
    MQHCONN   Hcon; /* connection handle */
    MQHOBJ    HobjInqNL; /* object handle */
    MQLONG    CompCode; /* completion code */
    MQLONG    Reason; /* reason code */
}

```



```

MQOD      InqObjDesc = {MQOD_DEFAULT};

MQLONGopenOptions, select[1], queueFound, count;

MQLONG    IAV[1];

PMQCHAR   pNames, pNamelist, pFirstBlank, pQueueStart;

pMQCXP->ExitResponse    = MQXCC_OK;
pMQCXP->ExitResponse2   = 0;
pMQCXP->Feedback        = 0;

/*****
/* Set the pointers to the exit work area.
*****/

*pExitBufferAddr      = (PMQBYTE16)pMQCXP->ExitUserArea;
*pExitBufferLength    = 16;

/*****
/* Determine the processing sequence based on the exit identifier
*****/

switch(pMQCXP -> ExitId)
{
    case MQXT_CHANNEL_MSG_EXIT:
        switch(pMQCXP->ExitReason)
        {
            case MQXR_INIT:
                if ((fp=fopen("c:\\qclamp.log","w")) == NULL)
                {
                }
            else
            {
                print_timestamp();
                fprintf(fp,"Msgexit init invoked\n");
                fclose(fp);
            }
            break;
        case MQXR_MSG:
            if ((fp=fopen("c:\\qclamp.log","a")) == NULL)
            {
            }
            else
            {
                print_timestamp();
                fprintf(fp,"Msgexit msg invoked\n");
                pMQXQH = (PMQXQH)pAgentBuffer; /* begin mapping message data */
            }
        }
}

```

```

print_timestamp();
fprintf(fp,"MsgUserData %.*s\n", 32, pMQCD->MsgUserData);
print_timestamp();
fprintf(fp,"Qmgr %.*s\n", 48, pMQCD->QMgrName);

MQCONN(pMQCD->QMgrName,      /* queue manager
*/
        &Hcon,              /* connection handle
*/
        &CompCode,         /* completion code
*/
        &Reason);          /* reason code
*/

print_timestamp();
fprintf(fp,"MQCONN ended with reason %ld\n", Reason);

InqObjDesc.ObjectType=MQOT_NAMELIST;
strncpy(InqObjDesc.ObjectName, pMQCD->MsgUserData,
sizeof(InqObjDesc.ObjectName));

openOptions = MQOO_INQUIRE;

MQOPEN(Hcon,                /* connection handle      */
        &InqObjDesc,       /* object descriptor for queue */
        openOptions,       /* open options          */
        &HobjInqNL,       /* object handle         */
        &CompCode,        /* MQOPEN completion code */
        &Reason);         /* reason code           */

print_timestamp();
fprintf(fp,"MQOPEN Namelist for INQ ended with reason %ld\n",
Reason);

select[0] = MQIA_NAME_COUNT; /* attribute selectors */

MQINQ(Hcon,                /* connection handle      */
        HobjInqNL,        /* object handle         */
        1L,               /* Selector count        */
        select,           /* Selector array        */
        1L,              /* integer attribute count */
        IAV,             /* integer attribute array */
        0,               /* character attribute count */
        NULL,            /* character attribute array */
        &CompCode,       /* completion code       */
        &Reason);        /* reason code           */

print_timestamp();

```

```

fprintf(fp,"MQINQ NL ended with reason %ld\n", Reason);
print_timestamp();
fprintf(fp,"Namelist entries %ld\n", IAV[0]);

pNames = (char*)malloc(IAV[0]*48);
print_timestamp();
fprintf(fp,"Memory allocated\n");

select[0] = MQCA_NAMES; /* attribute selectors */

MQINQ(Hcon,          /* connection handle          */
      HobjInqNL,     /* object handle          */
      1L,            /* Selector count         */
      select,        /* Selector array         */
      0L,           /* integer attribute count */
      IAV,          /* integer attribute array */
      IAV[0]*48,    /* character attribute count */
      pNames,       /* character attribute array */
      &CompCode,    /* completion code       */
      &Reason);    /* reason code           */

pNamelist = pNames;

print_timestamp();
fprintf(fp,"MQINQ NL ended with reason %ld\n", Reason);

print_timestamp();
fprintf(fp,"Namelist entries %.*s\n", IAV[0]*48, pNames);
print_timestamp();
fprintf(fp,"MQINQ NL ended with reason %ld\n", Reason);

count = 1;
queueFound = FALSE;

while(count <= IAV[0])
{
    if (!strncmp(pMQXQH->RemoteQName, pNamelist,
MQ_Q_NAME_LENGTH))
    {
        print_timestamp();
        fprintf(fp,"Found queue in namelist\n");
        queueFound = TRUE;
        break;
    }
    else
    {
        count++;
        pNamelist = pNamelist + MQ_Q_NAME_LENGTH;
    }
}

```

```

    }

    free(pNames);
    print_timestamp();
    fprintf(fp,"Memory freed\n");

    if (!queueFound)
    {
        pQueueStart = (PMQCHAR)pMQXQH->RemoteQName;
        pFirstBlank = strstr(pMQXQH->RemoteQName, " ");

        print_timestamp();
        fprintf(fp,"Queue name
length(%ld)\n",pFirstBlank-pQueueStart);

        if(pFirstBlank-pQueueStart <= 41 )
        {
            strncpy(pFirstBlank, ".SECURE", 7);
            print_timestamp();
            fprintf(fp,"Queue name changed, .SECURE added\n");
        }
        else
        {
            print_timestamp();
            fprintf(fp,"Queue name length(%ld) too long to add
\".SECURE\"\n",pFirstBlank-pQueueStart);
        }

    }

    MQCLOSE(Hcon,          /* connection handle          */
            &HobjInqNL,
            MQCO_NONE,
            &CompCode,
            &Reason
    );

    print_timestamp();
    fprintf(fp,"MQCLOSE NL ended with reason %ld\n", Reason);

    if (pDataLength != 0)
        dump_message(pAgentBuffer, pDataLength);

    fclose(fp);
}
break;

case MQXR_TERM:
    if ((fp=fopen("c:\\qc\\amp.log", "a")) == NULL)

```

```

        {
        }
        else
        {
            print_timestamp();
            fprintf(fp,"Msgexit term invoked\n");
            fclose(fp);
        }
        break;
    }
    break;

case MQXT_CHANNEL_SEND_EXIT:
    break;

case MQXT_CHANNEL_RCV_EXIT:
    break;

case MQXT_CHANNEL_SEC_EXIT:
    break;

default:
    break;
}
}

void dump_message(PMQBYTE pAgentBuffer, PMQLONG pDataLength)
{
#define CHARS_PER_LINE 16 /* Used in formatting the message */
extern FILE *fp;

char line_text[CHARS_PER_LINE + 1];
int chars_this_line = 0;
int lines_printed = 0;
int page_number = 1;
int InDataLen;
int ch;

InDataLen = *pDataLength;
ch = 0;

do
{
    chars_this_line = 0;
    print_timestamp();
    fprintf(fp,"%08X: ",ch);
    while ( (chars_this_line < CHARS_PER_LINE) &&
            (ch < InDataLen) )

```

```

    {
        if (ch % 2 == 0) fprintf(fp, " ");
        fprintf(fp, "%02X", pAgentBuffer??(ch??) );
        line_text??(chars_this_line??) =
            isprint(pAgentBuffer??(ch??)) ? pAgentBuffer??(ch??) : '.';
        chars_this_line++;
        ch++;
    }

    if (chars_this_line < CHARS_PER_LINE)
    {
        for ( ;chars_this_line < CHARS_PER_LINE; chars_this_line++)
        {
            if (chars_this_line % 2 == 0) fprintf(fp, " ");
            fprintf(fp, " ");
            line_text??(chars_this_line??) = ' ';
        }
    }
    line_text??(chars_this_line??) = '\0';
    fprintf(fp, "%s'\n", line_text);
    lines_printed += 1;
    if (lines_printed >= 60)
    {
        lines_printed = 0;
        fprintf(fp, "\f ");
    }
}

while (ch < InDataLen);
    /* end of print message loop */

}

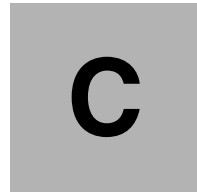
void print_timestamp(void)
{
    /* declares */
    extern FILE *fp;
    struct tm *ptr;
    time_t lt;
    char time_array[26];
    int buflen;

    lt = time(NULL);
    ptr = localtime(&lt);
    strcpy(time_array, asctime(ptr));
    buflen = strlen(time_array);
    time_array[buflen-1] = '\0';

    fprintf(fp, "<%s> ", time_array);
}

```

```
/* __FILE__ and __LINE__ C are C predefined macros */  
}
```

Additional information

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

<ftp://www.redbooks.ibm.com/redbooks/SG246814>

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246814.

How to configure HTTP Server Solution(iKeyman) certificate service

The iKeyman GUI illuminates keys to show you the password strength and we recommend that you use this tool as a guide to password strength. Here is an example showing a medium strength password that was used to create the key store.

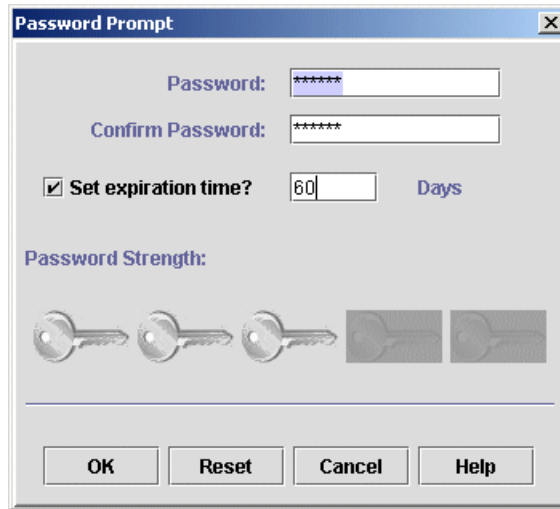


Figure C-1 Example showing medium strength password

Note: This password or pass phrase (which is another way of saying 'a sentence') now protects all of the keys maintained by iKeyman.

Once the key store is built, you can generate a key request. Behind the scenes, iKeyman generates a random number seed; this is something to start the process of creating a random number string. Seeds are usually generated by things such as mouse position, time between key clicks or some other fairly random event. Once a random number is generated, it is used to create an Asymmetric key pair; this is a Diffie-Hellman key pair named after the inventors of Private Key-Public key cryptography (Whitfield Diffie and Martin Hellman).

Many many books will explain asymmetric key pairs to you, Applied Cryptography by Bruce Schneier, being amongst the best. For the purposes of this redbook, all you need to know is that when you generate a certificate request, the request is encrypted with your private key (the one just generated by iKeyman) so that you have a unique request that will be forwarded to a Certificate Authority(CA) or Registration Authority(RA) for processing.

The CA processes the request, and creates a signed certificate that is sent back to you where it can be used in conjunction with your private key (which never left your machine) to sign things and to prove your identity (authentication and non repudiation) and to participate in encryption schemes that use symmetric key cryptography for privacy and integrity (so that nobody can see your message or change its contents).

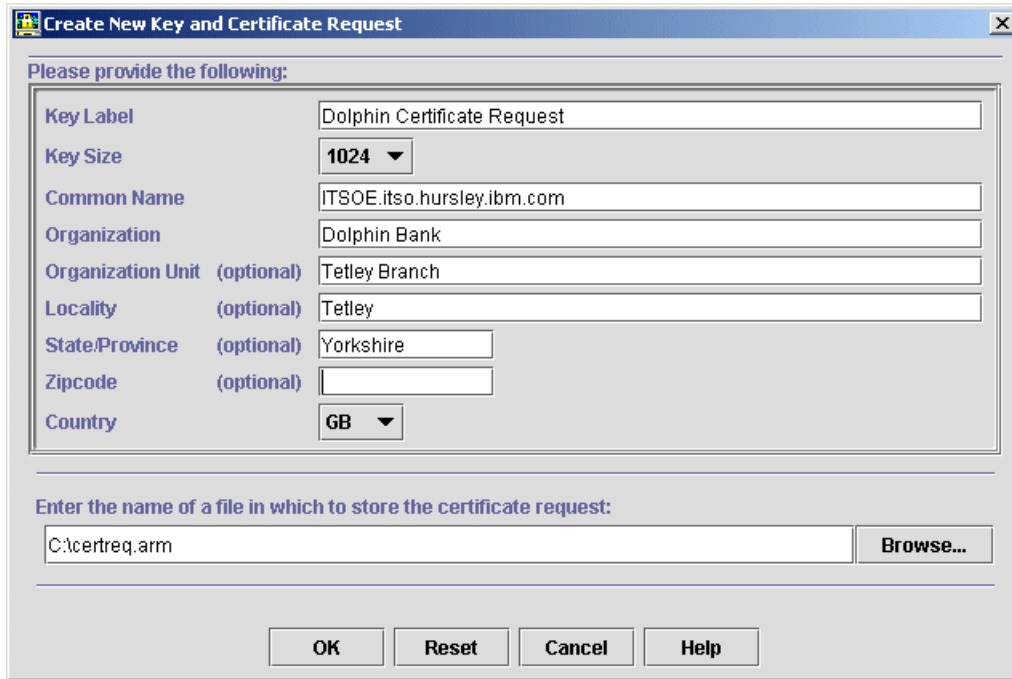


Figure C-2 Example showing how to generate a key request

This is what the certificate request that we just generated looks like in Base64 encoding;

```

-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBxzCCATAQAQAwYYxCzAJBgNVBAYTAkdCMRIwEAYDVQQIEw1Zb3Jrc2hpcmUx
DzANBgNVBACTB1R1dGx1eTEVMBMGAlUEChMMRG9scGhpbiBCYW5rMRYwFAYDVQQL
Ew1UZXRzZXkgQnJhbmNoMSMwIQYDVQQDExpJVFNP5pdHNvLmh1cnNsZXkuaWJt
LmNvbTCBnzANBgkqhkiG9w0BAQEFAA0BjQAwYkCgYEA2X1bMveThGRLJTk4FI/4
DCcyVuzC+FcQAIp9yejc27B698BK6/d8Ui828rkqRTBXQJ6810fni sexaHX771GJ
SHPerUp3zU2J9GGB3oEqvGw3t2IYL3LBucjtm1V1yHUSJ7FQg+8bx/Wn6vNGe8Df
FRY6yhw8TSt40v1V7IiFBxECawEAAaAAMAOGCSqGSiB3DQEBAUAA4GBAFr02nMa
3i1QLYm1Fow61JaTnodX1yTky6NCVw6cj5GfiEF8F1MSaouvCdRp9y9f9+E16RKV
wJfemIo04cxLvvo4Fffdw5WIpxMnJvPYP2PKeGS9HKWvN1IaerKpyPZBF9TDGkR
oVVuwrC1cajiwAmguGA0v5EXrxuRALZ+NEdI
-----END NEW CERTIFICATE REQUEST-----

```

Once you have a valid certificate request, it is sent to a CA or RA for processing. Remember that the certificate request is sent with the public key generated, the private key associated with the certificate request is stashed in a key repository until the signed certificate is returned.

For testing purposes, self-signed certificates can be generated by products such as iKeyman that bypass the request-signing process and effectively do both in a single execution.



Figure C-3 Example showing how to generate a self-signed certificate

This is what a self-signed certificate looks like in Base64 encoding:

```
-----BEGIN CERTIFICATE-----
MIICbzCCAdigAwIBAgIEPbUrUzANBgkqhkiG9w0BAQQFADB8MQswCQYDVQQGEwJH
QjESMBAGA1UECBMJWW9ya3NoaXJ1MQ8wDQYDVQQHEwZUZXRzZXkxFTATBgNVBAoT
DERvbHBoaW4gQmFuazEwMBQGA1UECXMNVGV0bGV5IEJyYW5jaDEZMBCGA1UEAxMQ
SW50ZXJ1ZXQgQWRkcmVzc2AeFw0wMjEwMjEwMDQxMjNaFw0wMzEwMjEwMDQxMjNa
MHwxZzAjbG9uYVYtYkdCMRlWZAYDVQQIEw1Zb3Jrc2hpcmUxZzANBgNVBAClTB1R1
dGx1eTEVMBMGA1UEChMMRG9scGhpbjBCYW5rMRYwFAYDVQQLEw1UZXRzZXkgQnJh
bmNoMRkwFwYDVQQDExBJbnR1cm51dCBZGRyZXNzMIGfMAOGCSqGSIb3DQEBAQUA
A4GNADCBiQKBgQDRaZHRcC/Gpz11ReRp79xHAzPwRrBvbYAoXOH56yjeaZNEAyn
iN6sfPbR6QBw1ItsmUXsjqm8FfG8D/GtEwX9WkLPWe4bC229Ww9rJj1+dMewK1m1
UdG0/Va5M63qjwWSUbmUGZCeVA2yhSuxQmn3afKhFHOAewJjtOMnSSwiWIDAQAB
MAOGCSqGSIb3DQEBAUAA4GBADjQ1q88ZCIQBVA7zEcnrLqNac5Q4D7tK6WCfOE
Px10oxL03K3gNWWAvhgJMHZ9F8Fy6gzpRMVYdTx/yRpFCf0603GtGhRqbcRaZmM
RbKrAtAU9NzWuIWuSuAnUqVGspvNQwwGueTijjy53d4vWpPQQasymPdbmv+mkNyoG
bEIG
-----END CERTIFICATE-----
```

Notice that this certificate is bigger than the request and that is because it contains the public key of the entity that signed the certificate.

Certificate request processing

Many customers will wish to automate the process whereby certificate requests are processed by a CA or RA. Such processes can be easily automated when certificates are just being re-issued to change the expiry date, but for first time processing, the process really needs to involve human interaction.

Customers may wish to explore using the IBM Business Process Manager (formerly MQSeries Workflow) product to automate this process.

How to configure Microsoft Windows 2000 certificate services

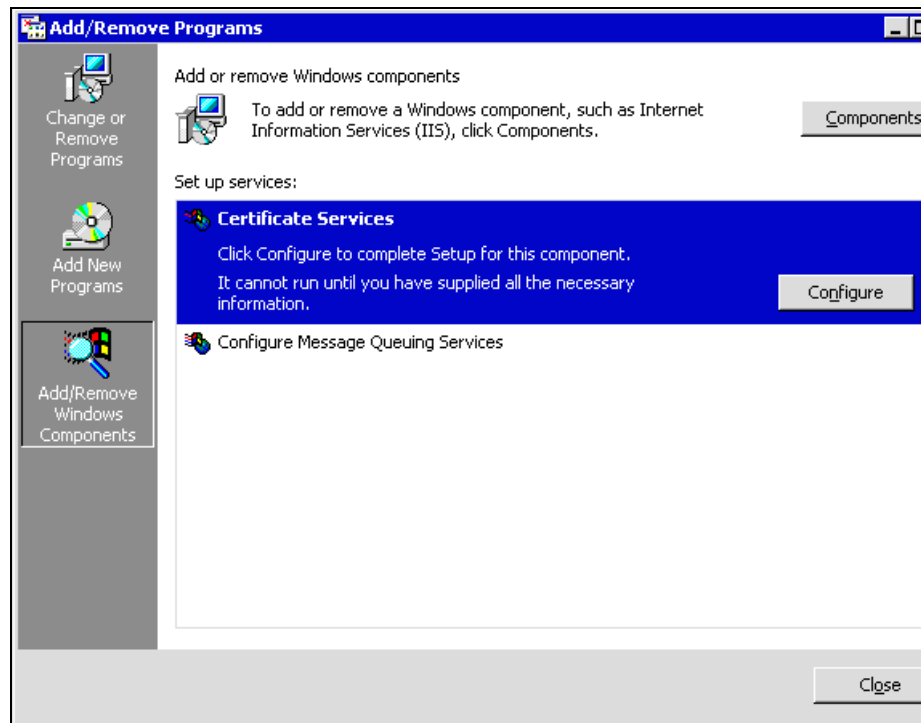


Figure C-4 Add/remove Windows components

Install with Windows 2000 certificate services

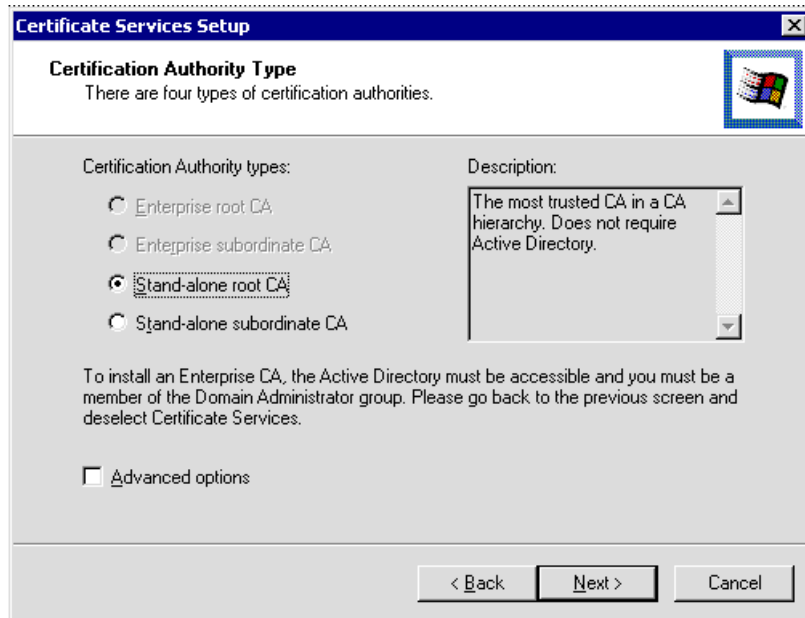


Figure C-5 Certificate services setup

If you have not setup active directory only stand alone will be allowed

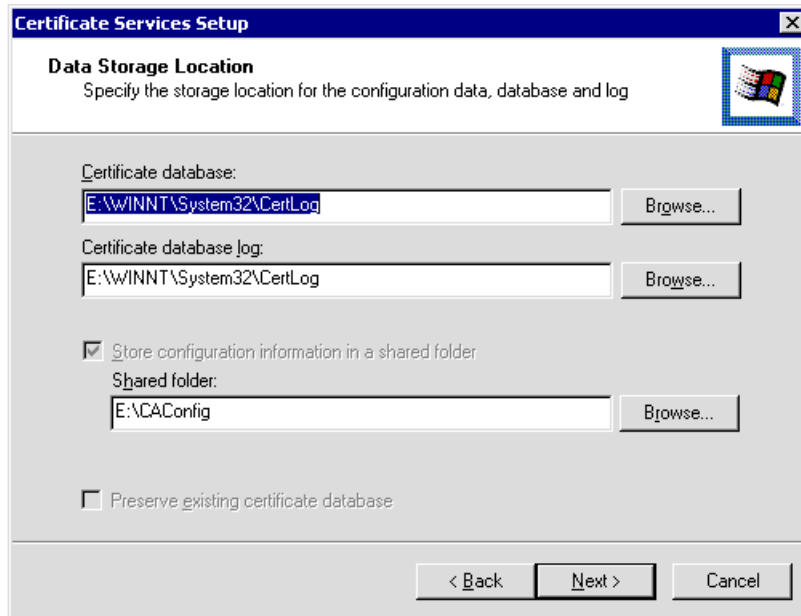


Figure C-6 More setup

After completing the certificate information, protect the areas where keys will be stored. The root CA certificate will automatically be written to the shared folder.

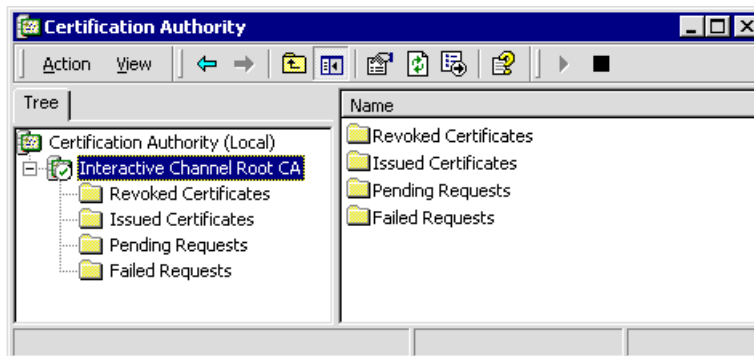


Figure C-7 CA MMC

Certificate services are managed with MMC and can be started from Start-> Program Files -> Admin Tools

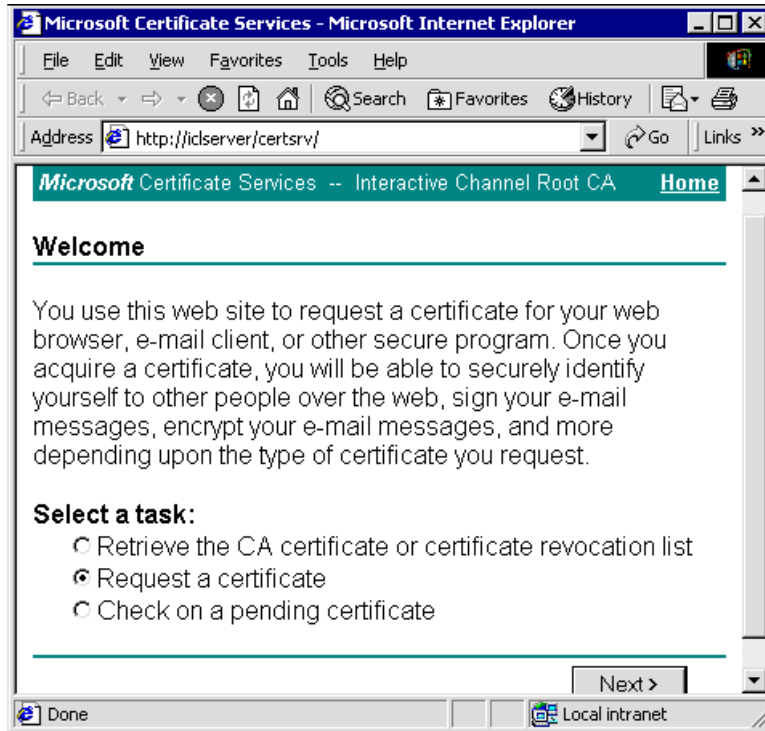


Figure C-8 Certificate server

Connect to the certificate server by entering `http://servername/certsrv` in a browser from any machine on the same network

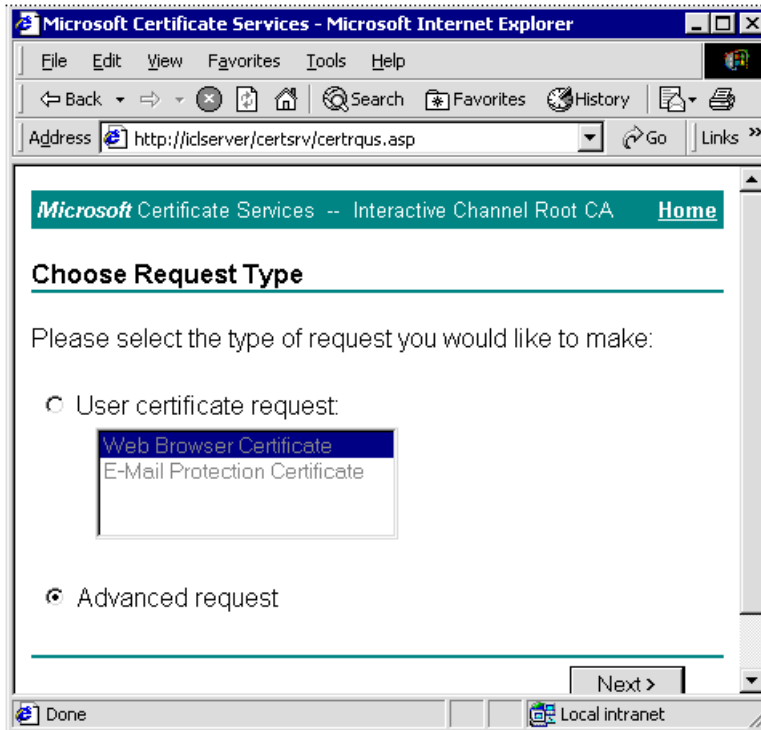


Figure C-9 Certificate request

Select advanced request for a queue manager certificate.

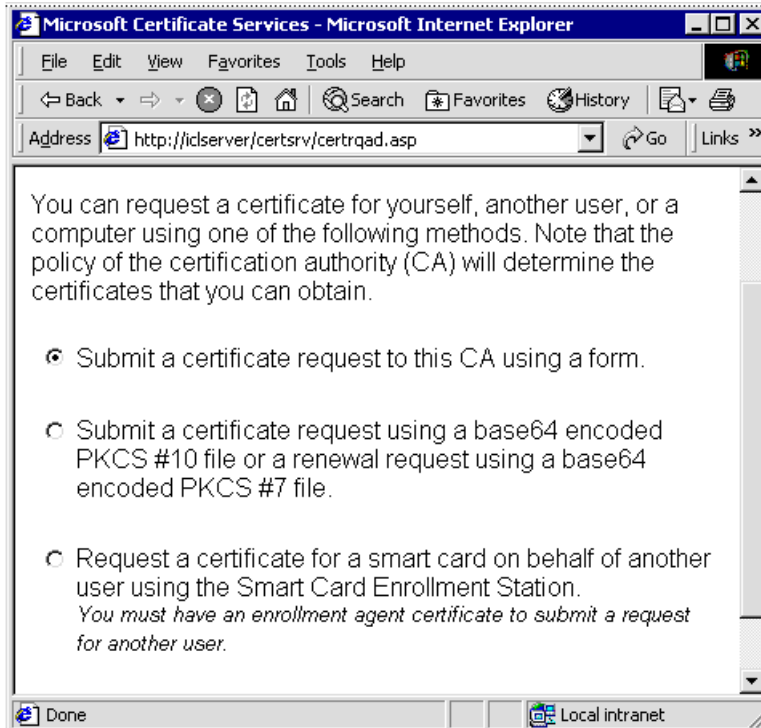


Figure C-10 Certificate type

Complete the form for the certificate server to generate the keys.

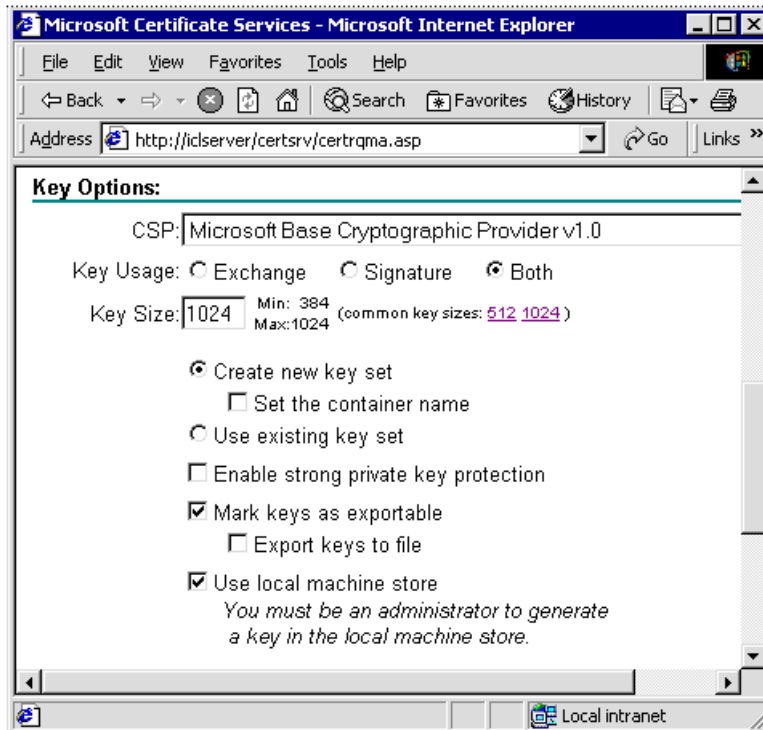


Figure C-11 Key options

Complete the form which is similar to a CA form, paying attention to the key options if you intend to move the keys from the machine you are requesting from.

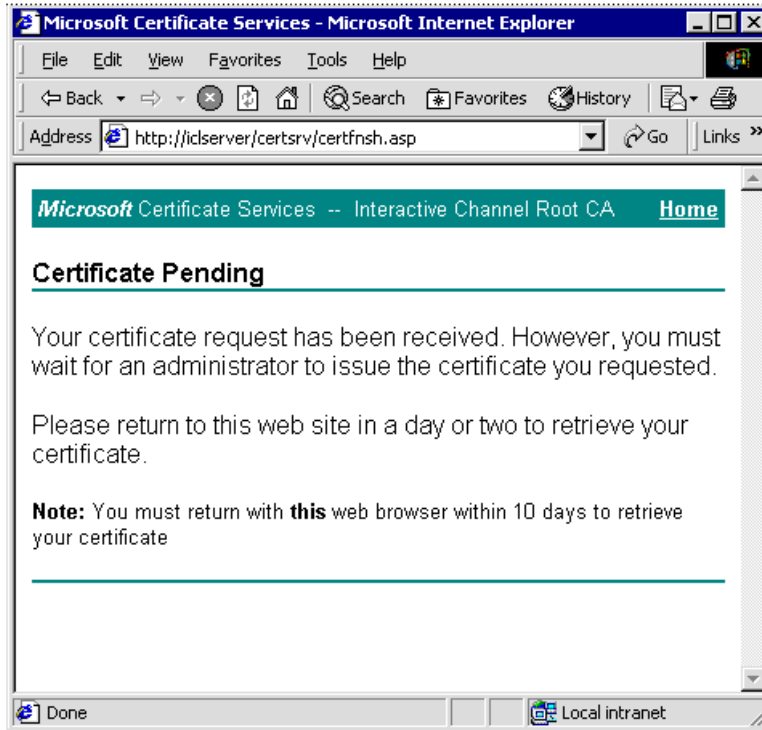


Figure C-12 Pending request

Depending on how the certificate server is set up, it may be necessary to wait for the administrator to approve the request.

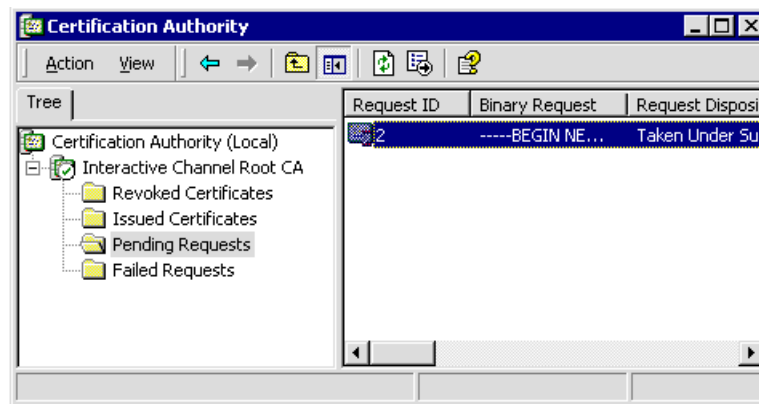


Figure C-13 CA pending requests

In the MMC the certificate awaits approval. Right-click all tasks as shown above.

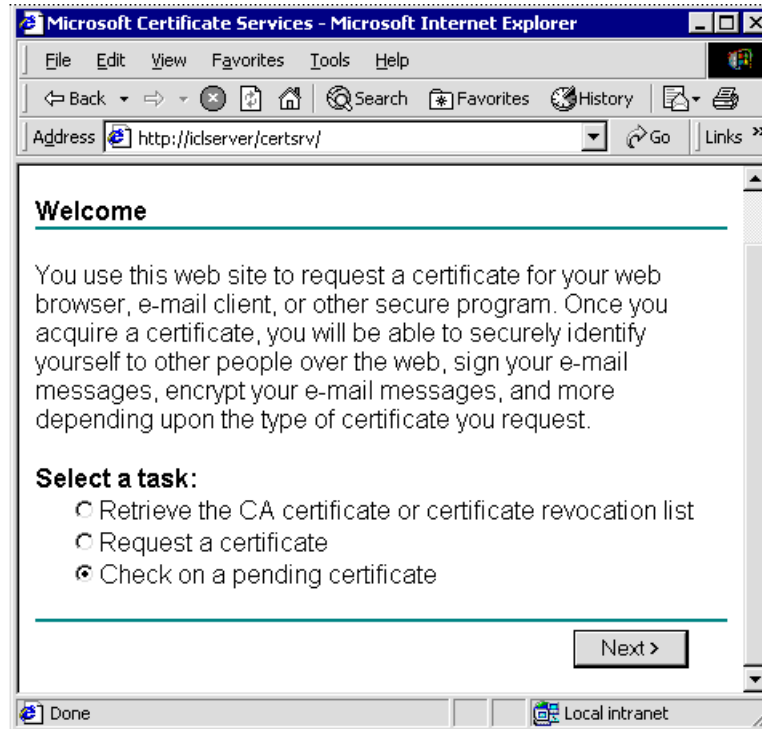


Figure C-14 Certificate collections

The certificate can be collected in the Web browser. Follow the on-screen instructions.

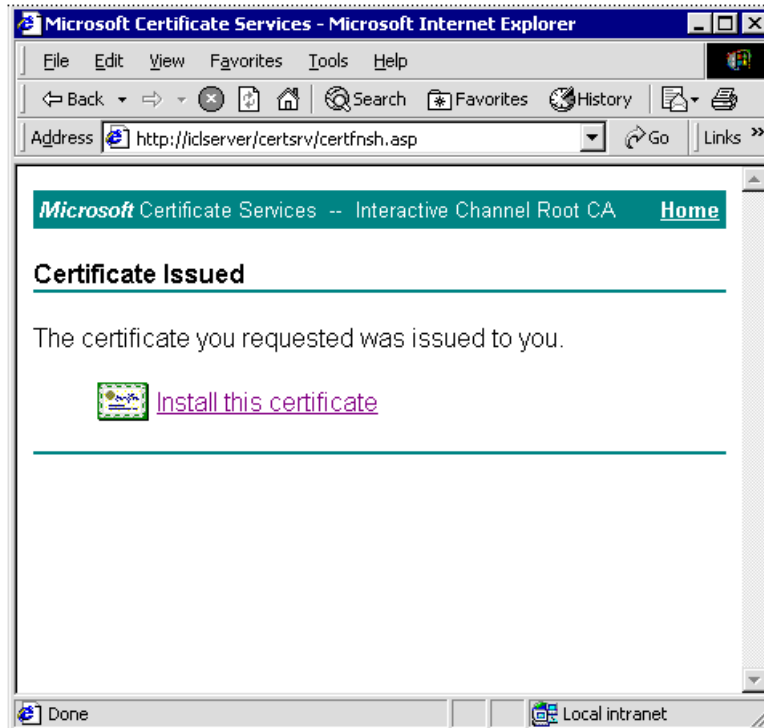


Figure C-15 Install certificate

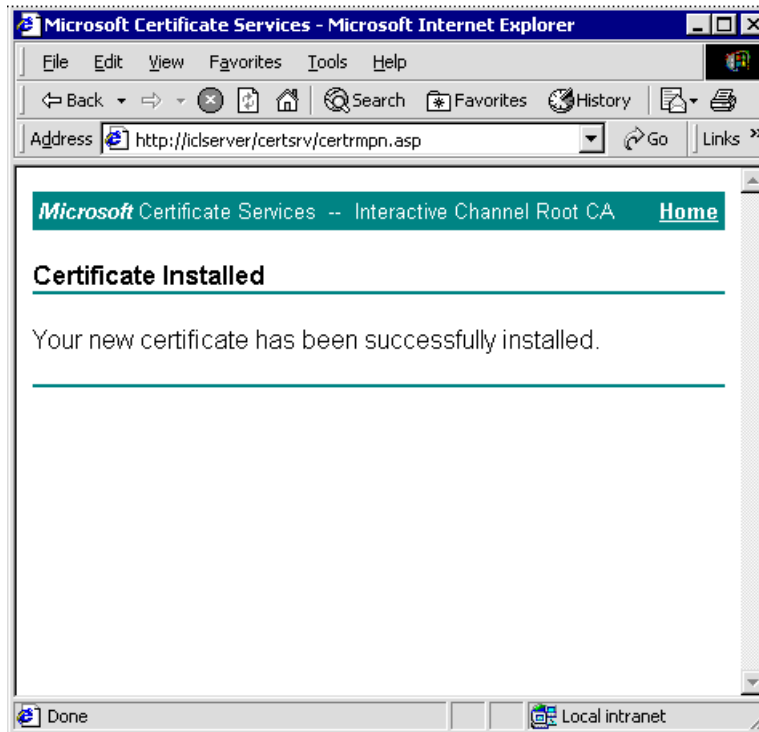


Figure C-16 Certificate successfully installed

How to use OpenSSL

The arguments for using OpenSSL and creating self-signed certificates for each queue manager and client are strong. When a CRL is not in use, if one of the private keys is compromised, then all of them are affected. When self-signed certificates are used, if a private key is compromised, only the connecting clients and queue managers are affected. This is a much smaller overhead, since typically, not all queue managers communicate with all other queue managers.

The source code for OpenSSL is available from:

<http://www.openssl.org/source>

Once openssl.exe is compiled (see the OpenSSL INSTALL.W32 guide for instructions), only two further commands need to be issued to generate a PKCS12 certificate. First of all, it is necessary to create a configuration file, which is used by the first command.

```

prompt = no
[ req ]
default_bits = 1024
distinguished_name = req_distinguished_name
[ req_distinguished_name ]
countryName = GB
stateOrProvinceName = Dorset
localityName = Poole
organizationName = Interactive Channel Ltd
organizationalUnitName = WebSphere MQ Certificates
commonName = QMGR
emailAddress = info@interactive-channel.com

```

Figure C-17 *openssl.tmp*

The two commands can now be issued either at the command line or from a batch file.

```

Microsoft Windows 2000 [Version 5.00.2195]
<C> Copyright 1985-2000 Microsoft Corp.

C:\>certs.bat

C:\>openssl req -config openssl.tmp -new -x509 -outform PEM -out cert.pem -keyout
key.pem -keyform PEM -sha1 -days 730 -passout pass:password
Using configuration from openssl.tmp
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.+++++
writing new private key to 'key.pem'
-----

C:\>openssl pkcs12 -export -in cert.pem -inkey key.pem -out key.p12 -passin pass
:password -passout pass:password
Loading 'screen' into random state - done

C:\>del cert.pem
C:\>del key.pem
C:\>_

```

Figure C-18 *certs.bat*

You will notice in the batch file above that the PEM keys are deleted after the certificate has been created. This prevents them from being reused.

The certificate can now be imported into Internet Explorer with a private key exportable set. It can then be added to a default store queue manager and assigned to the queue manager. See the *WebSphere MQ Security* manual for further details.

The public certificate needs to be distributed to communicating queue managers and clients. This can be achieved by exporting the public certificate in Internet Explorer, without a private key, in base64 encoded x.509. The certificate with the private key can then be deleted from the Internet Explorer store.

Glossary

AIX IBM version of UNIX operating system. AIX is multi-user system, the same as other UNIX system.

Attribute Authority (AA) An authority trusted by one or more users to create and sign attribute certificates. It is important to note that the AA is responsible for the attribute certificates during their whole lifetime, not just for issuing them.

Attribute Certificate (AC). A data structure containing a set of attributes for an end-entity and some other information, which is digitally signed with the private key of the AA which issued it.

Certificate Can refer to either an AC or a public key certificate. Where there is no distinction made the context should be assumed that the term could apply to both an AC or a public key certificate.

Certification Authority (CA) An authority trusted by one or more users to create and assign public key certificates. Optionally, the CA may create the user's keys. It is important to note that the CA is responsible for the public key certificates during their whole lifetime, not just for issuing them.

Certificate Policy (CP) A named set of rules that indicates the applicability of a public key certificate to a particular community or class of application with common security requirements. For example, a particular certificate policy might indicate applicability of a type of public key certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.

Certification Practice Statement (CPS) A statement of the practices which a CA employs in issuing public key certificates

End-entity A subject of a certificate who is not a CA in the PKI or an AA in the PMI. (An EE from the PKI can be an AA in the PMI.)

IBM Business Process Manager Formerly IBM MQSeries Workflow.

Information security A process of securing a system known as security engineering. A process of negating risk and balancing the cost of a security solution implementation and subsequent support with the business rewards.

MCAUSER ID A field in the channel definition where a unique user ID can be set that has authority to connect to a queue manager.

PKCS#7 data format standard This standard specifies that the symmetric encryption key generated to encrypt the message will itself be encrypted using the public key assigned to the target application. This allows the encryption key to be passed along with the data in a secure format avoiding the need to do some form of "out of band" key exchange. It also ensures that no one can decrypt the message without the private key of the target application.

Public Key Infrastructure (PKI) The set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke PKCs based on public-key cryptography.

Privilege Management Infrastructure (PI)

A collection of ACs, with their issuing AA's, subjects, relying parties, and repositories, is referred to as a Privilege Management Infrastructure.

Registration Authority (RA) An optional entity given responsibility for performing some of the administrative tasks necessary in the registration of subjects, such as: confirming the subject's identity; validating that the subject is entitled to have the values requested in a PKC, and verifying that the subject has possession of the private key associated with the public key requested for a PKC.

Relying party A user or agent (for example, a client or server) who relies on the data in a certificate in making decisions.

Root CA A CA that is directly trusted by an EE; that is, securely acquiring the value of a root CA public key requires some out-of-band step(s). This term is not meant to imply that a root CA is necessarily at the top of any hierarchy, simply that the CA in question is trusted directly. Note that the term "trust anchor" is commonly used with the same meaning as "root CA" in this document.

Subordinate CA A "subordinate CA" is one that is not a root CA for the EE in question. Often, a subordinate CA will not be a root CA for any entity but this is not mandatory.

Subject A subject is the entity (AA, CA, or EE) named in a certificate, either a PKC or AC. Subjects can be human users, computers (as represented by Domain Name Service (DNS) names or Internet Protocol (IP) addresses), or even software agents.

TCB Trusted computer base

Time Stamp Authority (TSA) A TSA is a trusted Third Party who provides a "proof-of-existence" for a particular datum prior to an instant in time.

Top CA A CA that is at the top of a PKI hierarchy. Note: This is often also called a "Root CA," since in data structures terms and in graph theory, the node at the top of a tree is the "root." However, to minimize confusion in this document, we elect to call this node a "Top CA," and reserve "Root CA" where there is a single CA directly trusted by the EE. Readers new to PKIX should be aware that these terms are not used consistently throughout the PKIX documents, as the Internet PKI profile [2459bis] uses "Root CA" to refer to what this and other documents call a "Top CA," and "most-trusted CA" to refer to what this and other documents call a "Root CA."

Abbreviations and acronyms

B2B	Business to Business	DES	Data Encryption Standard
ACE	Access Control Entities	DLQ	Dead Letter Queue
ACF2	Access Control Facility 2	DN	Distinguish Name
ACL	Access Control Lists	DOS	Denial of Service
ALE	Annual Loss Expectancy	DR	Disaster Recovery
AMBI	Access Manager for Business Integration	ECC	Elliptical Curve Cryptography
AMI	Application Message Interface	EFS	Encrypted File System
API	Application Programming Interface	ESM	External Security Manager
APPC	advanced program-to-program communication	FTP	File Transfer Protocol
AUP	Acceptable Use Policy	GC	Global Catalog
CA	Certificate Authority	GINA	Graphical Identification and Authentication
CF	Couple Facility	GSS	Generic Security Service
CHINIT	Channel Initiator	GUI	Graphical User Interface
CICS	Customer Information Control System	HLQ	High Level Qualifier
CNF	Certificate Name Filtering	IBM	International Business Machines Corporation
CORBA	Consultive, Objective and BI-functional Risk Analysis	ICSF	Integrated Cryptographic Service Facility
CRAMM	Central Computer and Telecommunications Agency Risk Analysis and Management Method	IGQ	Intra Group Queueing
CRL	Certificate Revocation List	IKE	Internet Key Exchange
DAACL	Discretionary Access Control List	IMS	
DB2	Database 2™ (an IBM relational database management system)	ITIM	IBM Tivoli Identity Manager
DCE	Distributed Computing Environment	ITSO	International Technical Support Organization
		JMS	Java Message Service
		LAN	Local Area Network
		LDAP	Lightweight Data Access Protocol
		LPAR	Logical Partition
		LSA	Local Security Authority
		LUID	Locally Unique Identifier
		MAC	Message Authentication Code

MQI	Message Queue Interface	SOCKS	Software Common Knowledge IR System
MQMD	MQ Message Descriptor	SRM	Security Reference Monitor
NFS	Network File System	SSL	Secure Sockets Layer
NIST	USA National Institute of Standards	SSPI	Security Support Provider Interface
NSA	National Security Administration	TCA	Task Control Area
NTLM	NT Lan Manager	TCB	Trusted Computing Base
OAM	Object Authority Manager	TCP/IP	Transmission Control Protocol/Internet Protocol
OAM	Object Authority Manager	TLS	Transport Level Security (the new name for SSL)
OCSP	Online Certificate Status Protocol	TSO	
OS ID	Operating System ID	UDB	Universal Database
PCF	Programmable Command Facility	VIPA	Virtual IP Address
PGP	Pretty Good Policy	VPN	Virtual Private Networks
PKA	Public Key Algorithm	WAP	Wireless Access Points
PKI	Public Key Infrastructure	WEP	Wireless Equivalent Privacy
QMGR	Queue Manager	XCF	Cross Coupling Facility
QSG	Queue Sharing Group		
RA	Registration Agency		
RACF	Resource Access Control Facility		
RSA	Rivest-Shamir-Adleman algorithm (cryptograph, named after Ronald Rivest, Adi Shamir and Leonard Adleman)		
SACL	System Access Control List		
SAF	System Authorization Facility		
SAK	Secure Attention Key		
SAM	Security Accounts Manager		
SAPI	Security Application Programming Interface		
SDK	Software Development Kit		
SETTM	Secure Electronic TransactionTM		
SID	Security Identifier		

Related publications

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

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 414.

- ▶ *An Implementation Guide for AS/400 Security and Auditing: Including C2, Cryptography, Communications, and PC Connectivity*, GG24-4200-00
- ▶ *AS/400 Tips and Tools for Securing Your AS/400*, SC41-5300-04
- ▶ *OS/400 Security - Reference V4R5*, SC41-5302-04
- ▶ *AS/400 Internet Security: Protecting Your AS/400 from HARM in the Internet*, SG24-4929-00
- ▶ *AIX 5L and Windows 2000: Side by Side*, SG24-4784-02
- ▶ *AIX 4.3 Elements of Security Effective and Efficient Implementation*, SG24-5962-00
- ▶ *Additional AIX Security Tools on IBM eServer pSeries, IBM RS/6000, and SP/Cluster*, SG24-5971-00
- ▶ *WebSphere MQ for z/OS System Setup Guide*, SC34-6052
- ▶ *Using MQ on AS/400*, SG24-5236-00
- ▶ *iSeries System Administration Guide*, SC27-1136-00

Other resources

These publications are also relevant as further information sources:

- ▶ *RFC 2196, Site Security Handbook*

Referenced Web sites

Relevant web sites are listed in respective chapters of this book.

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the **CD-ROMs** button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

Symbols

"mqm" 54
"mqm" group 54
'key strength' 39
"Central Computer and Telecommunications Agency Risk Analysis and Management Method" (CRAMM) 23
"Consultative, Objective and Bi-functional Risk Analysis" (COBRA) 23
"Security policy" 23
"waterfall" life cycle model used in software engineering 12
"waterfall" model 10
'z/OS Security Server' 6

A

A technical look at AMBI 116
Access control 101
Access Control Entities (ACE) 60
Access Control List (ACL) 60
Access control list (ACL) 88
Access Control Lists (ACL) 53
access management solutions 116
Access Manager family 110, 116
Access Manager for Business Integration (AMBI) 99
Access Manager for Business Integration 4.1 125
Access Manager for Business Integration interceptor and server (daemon) components 122
Access Manager for Business Integration server (daemon) 121
Access Manager for Business Integration server (daemon) components 123
Access Manager Policy Server 117
Access Manager runtime component (AMRTE or pdrte) 118
Access Manager runtime libraries 117
Access to binaries (executables) 190
Access to certificates 189
Access to commands 197
Access to files 189
Access to WebSphere MQ object definitions 190
action 5

active directory 58, 394
Additional logic 179
Administration 197, 199–201
administration tool 110
AES 37–38, 185, 191
Alias queues 198
Aliases 263
alternate user 262
ALTERNATE USER ID 160
Alternate user ID security 199
AM Authorization Services 124
AMB resources 125
AMBI futures 126
AMQCHLCL.TAB 241
AMQCLCHL.TAB 203
Annual Loss Expectancy (ALE) 22
API security 76, 198
APPC channel 263
application API calls 108
application level 116
Application Message Interface (AMI) 101
aspects to setting up a WebSphere MQ client to use SSL 203
aspects to setting up a WebSphere MQ queue manager 201
Assessing loss expectancy 22
assets 16
assigning rights 260
asymmetric key 390
attack simulations 132
attack trees 1
Attack types 186
Attribute Authority (AA) 6
Attribute Certificate (AC) 6
Audit 102
audit options 112
audit trail 62
Auditing 62
auditing 62
auditing categories 62
auditing policy 62
AUTHINFO 202
automate 393
aznAPI application 124

B

Basic isolation functions 46
basic WebSphere MQ security issues 99
Blank user IDs 262
Bruce Schneier (author of Secrets and Lies ISBN 0-471-25311-1) 5
Business to Business (B2B) 24

C

CA signed certificate 42
CA using Microsoft Windows 2000 certificate 33
central security policy manager 109–110
Certificate 6
certificate 32
Certificate authority 197
Certificate Authority (CA) 62, 390
Certificate holders 7
certificate management 32
certificate name filtering (CNF) 83
Certificate Policy (CP) 6
certificate request 390
Certificate request processing 393
Certificate Revocation List (CRL) 38
Certificate revocation lists 134
Certificate Revocation Lists (CRLs) 4, 7
Certification Authorities (CAs) 7
Certification Authority (CA) 6
Certification Practice Statement (CPS) 6
CF list structures 69
channel attribute 168
Channel definitions 199
channel initiator (CHINIT) 64
channel initiator address space (CHINIT) 64
Channel security 88, 102
channel security issues 105
checksum 36
CHINIT 234
CHINITs 263
CICS trigger monitor 177, 350
cipher 196
cipher policy 200
CipherSpec 36, 184
CipherSpec for WebSphere MQ 186
CipherSuite 35
classes of service 146
Client (Windows) 241
Client connections 200–201
client definition 241

Clients 7
CLISTs 168
CLNTCONN 203
CLNTCONN channel 241
cluster channel 202
clustering technology 5
clusters 142
Code 265
Command and command resource security 79
command line interface 110
command resource security 198
command resource security checking 67
Command security 67
Command server queues 261
common threats to WebSphere MQ security 17
components of Access Manager for Business Integration 121
concepts in Access Manager administration 120
confidential 196
confidentiality 116
Connection security 76, 93
Context Security 198
correct KEYRING 238
correct level of SSL support 235
coupling facility (CF) 68
Create a key repository 216
Creating the CA certificate 224
CRL 39
CRL check 202
Cross Coupling Facility (XCF) 70
Cryptoanalysis 39
cryptoanalysis 196
cryptographic algorithms 31, 35
cryptographic concepts 2
cryptographic co-processor 33, 191
cryptographic co-processor card 34
cryptographic hardware 204
cryptographic processors 191
cryptography libraries 111
CryptoHardware 204
CSD 146
CSQ6SYSP 261
CSQOPR 261
CSQUTIL 313
cyber attacks 1

D

Data Certification 6

- data flow 101
- data protection policies 112
- Data security 102
- data security issues 104
- database 38
- DCE security 88
- DCM 33
- Dead Letter Queue (DLQ) 261
- Decryption 17
- default objects 260
- default user ID 261
- DEFINE commands 313
- defining channels 261
- Denial of service 186
- Denial Of Service (DOS) 19, 134
- DES 34, 37, 184, 191
- detection 5, 196
- different permissions 263
- different switches 67
- digital certificate 5
- digital certificate management tool 41
- Digital Certificate Manager (DCM) 249
- directory 38
- Disaster Recovery (DR) 132
- Discretionary Access Control List (DACL) 60
- DISPLAY commands 313
- Distinguish Name (DN) 105
- distinguished name (DN) 109
- DN filtering 204
- DSA 34
- Dynamic queues 77

E

- Elliptical Curve Cryptography (ECC) 35
- Encrypted File System (EFS) 61
- End-entity 6
- Enterprise Security Manager (ESM) 101
- Event Viewer 62
- examples of security policy components 26
- Export the CA signed certificate to the other machines 215
- Exporting the signed certificates 233
- External Security Manager (ESM) 63

F

- Fault Tree Analysis (FTA) 19
- File Transfer Protocol (FTP) 139
- Flooding 18

- Fraud 187

G

- GENCERT command 224
- generic profile 91, 102
- generic profiles 263
- Generic Security Service (GSS) 88
- Generic switch profiles 66
- Global Catalog (GC) 59
- good guidelines for developing computer security policies 27
- good security practices 259
- Graphical Identification and Authentication (GINA) 57
- Group policies 60
- GSKit toolkit 34

H

- hardware 146
- Hash 182
- high performance 116
- HTTP server solution (iKeyman) 32

I

- IBM Directory Server 110, 117
- IBM LDAP Directory 110
- IBM Redbooks Web server 389
- IBM Tivoli Access Manager for Business Integration 107, 116
- IBM Tivoli Access Manager for Business Integration server (daemon) 109
- IBM Tivoli Identity Manager (ITIM) 110
- IBM z/OS Security Server 33
- ICSF 35
- iKeyman 32
- iKeyman GUI 33, 389
- Impersonation 17
- implementing the policy 28
- Information security 14
- Integrated Cryptographic Support Facility (ICSF) 191
- interceptor 108
- interceptor layer 108
- Internet explorer 404
- Intra Group Queueing (IGQ) 76
- iSeries 5, 149
- isk assessment 196

ISPF panels 263
issues 103

J

J2EE applications 119
Java Message Service (JMS) 101
JCL 265

K

key repository 41, 204, 391
key request 390
key store 390
Keys 183

L

LDAP 38–39
LDAP access 190
LDAP calls 39
LDAP compliant directory 117
LDAP directory 109, 134
LDAP server 39, 125
Life cycle models 9
Lightweight Data Access Protocol (LDAP) 4
limitations 126
linear logging 262
Local Security Authority (LSA) 57, 60
Logical Partitions (LPARs) 137

M

Make your keyring on z/OS 235
MAKEDEF 313
Making your own certificates 234
management tasks 135
map.conf file 124
MCAUSER 261
MD2 34
MD5 34, 186
MD5 Algorithm 36
medium strength password 389
Message Authentication Code 36
message channel agent (MCA) 101
Message exit 97
message flow 146, 159
message flows 153
message integrity 116
Message Queue Interface (MQI) 100
Message security 199

Messages 200–201
methods of WebSphere MQ client channel setting 203
methods used to model threats 19
modelling 13
MOVER 64
MQ CipherSpec 183
MQADMIN class 197
MQCD V7 204
MQCMD class 197
MQCONN class 198
MQCONN 241, 315
MQCONN API 203
mqm group 261
MQMD 78, 101, 262
MQSC command 238
MQSCO 204
MQSERVER 203
MQSNOAUT environment variable 262
MQSSLKEYR 241
MQSSLPEER 105
myths 14

N

NAMELIST 202
Native WebSphere MQ security issues 101
Native WebSphere MQ Security vs. AMBI Security 127
Netscape/iPlanet LDAP Directory 110
Network level security 47
network security 54
new security capabilities 137
NFS security 55
NT Lan Manager (NTLM) 56
NTFS 61

O

OAM 3, 168–169, 260
OAM administration commands 90
object 50
Object Authority Manager (OAM) 3, 101
object authority manager (OAM) 88
object authorization on OS/400 170
object definitions 146
Object security 50
Obtain a CA signed certificate 209, 216
Online Certificate Status Protocol (OCSP) 4
OpenSSL 33, 403

operating system ID (OS ID) 109
Organizational Registration Authorities 7
OS/400 security 50
Overview 1

P

parallel sysplex 5
Password attacks 188
password strength 389
Passwords 39
PCI board 34
pdmqazn.conf configuration file 124
persistent 103
Physical and procedural security 51
Physical security 50
PKCS#11 34
PKCS#7 data format 114
PKI 61, 112
PKI certificate management 134
PKI certificates 59, 61
Planning 1, 205
planning and this is described in the next chapter of this redbook. 5
Planning for SSL 40
policies 59
policy 132
policy development 13
policy document 196
policy enforcers 118
policy goals 28
policy implementation 13, 27
policy implementation document 28
Policy Server 117
portal applications 119
Prepare a certificate 206, 216
Prepare certificates 40, 224
Pretty Good Policy (PGP) 20
principal 88
private key 114, 197, 390–391, 403–404
Private key storage 197
Privilege Management Infrastructure (PMI) 7
Problem solution 178
procedures 45
process modelling 13
processes 10
Processing logic 178
profile 63
profiles 197

protected resource 63
Protection 5
pSeries 5, 148
public certificate 405
public key 114, 391
Public Key Certificate (PKC) 6
Public Key Infrastructure 6
Public Key Infrastructure (PKI) 3, 6, 32
Public key infrastructure (PKI) 7
Public Management Infrastructure (PMI) related 6

Q

QMQMADM 199
queue manager 68
queue naming conventions 260
queue sharing group 137, 263
queue sharing groups 5, 68

R

RACDCERT 224
RACF (Resource Access Control Facility) 6
RACF classes 63
RACF panels 34
random number seed 390
RC2 37
RC4 37–38, 185, 191
reaction 196
receive exit 97
Redbooks Web site 414
 Contact us xvii
Registration Authority (RA) 7, 390
Relying party 7
Repositories 7
Request flow 154
RESLEVEL 263
Reslevel security 199
Resource Access Control Facility (RACF) 34, 48, 63
Resource security 77
Response flow 155
restricted authority 261
Restricted data 197
RFC2196 196
risk 2
Risk assessment 9
risk assessment 13
risk assessment methods 22
risks 196

Root CA 7
RSA 34
runmqsc command 91

S

sample AMQSGET4 314
scalability 116
Scripts 265
secPKIMap 124
secret key 182
Secure Socket Layer (SSL) 3, 5
Secure Sockets layer (SSL) 83
secure WebSphere MQ infrastructures 31
Secureway Security Server for z/OS 48
securing an enterprise 45
Security Accounts Manager (SAM) 57
Security commands 87
security concerns 99, 177
Security definitions 9
security descriptor 60
Security Engineering 11
Security engineering 12
security engineering 29
security exit 96
security exposures 131, 259
security for z/OS WebSphere MQ 63
security goal 196
security goals 196
Security Identifier (SID) 58
security infrastructure 134
security log 62
security on AIX 200
security on OS/400 50, 199
security on Windows 2000 200
security on z/OS 197
security planning 131
security policies 109, 196
security policy 15, 25, 28, 116, 131, 196
security policy data 109
Security policy development 9
security policy document 27
Security policy implementation 9
security policy manager 109
security profiles 168
security protection 196
security reference monitor 60
Security Reference Monitor (SRM) 60
security risks 39
security set up on z/OS 168
Security setup 205
security solution 15, 46
Security solutions 15
Security Support Provider Interface (SSPI) 57
security technologies 2
Security terminology 6
security tools 196
self signed certificates 392
self-signed certificates 42, 403
send exit 97
sensitive 196
sensitive data 107
server applications 109
server definition 241
Setup for client (Windows) 241
SHA-1 34, 186
SHA-1 algorithm 36
shared queue definitions 68
shared queues 68
signed certificate 390–391
Signing certificate requests 225
SMF records 106
Sniffing 17
sniffing 183
software 146
Software Development Kit (SDK) 32
Software engineering 10
Solving the issues with AMBI 107
source code for OpenSSL 403
specific security 2
SSL 31, 137
SSL CipherSpec 204
SSL function 40
SSL in WebSphere MQ v5.3 181
SSL support 187
SSL technology 201
SSL.CLCHL 242
SSLCIPH 241
SSLKEYR attribute 238
SSLPeerNameLength 204
SSLPeerNamePtr 204
standard MQI library 108
standard WebSphere MQ API library 109
Store your certificate 210, 216, 222
Subject 7
Subordinate CA 7
suggested elements of policy implementation documents 28

- Summary of native WebSphere MQ security 106
- Summary of what AMBI provides 116
- Summary of when to use AMBI 127
- support pac MQSeries administration wrapper (MS0E) 91
- SVRCONN channel 241–242
- SVRCONN for client connection attributes for SSL 202
- Switch profiles 65
- symmetric cryptography services 111
- symmetric encryption key 114
- symmetric key cryptography 390
- Symmetric keys 37
- symmetrical key algorithm 191
- System Access Control List (SACL) 60
- System security 50–51
- System validation, maintenance and management 28
- System-level security 47

T

- Task Control Area (TCA) 178
- TCB 55
- TCP/IP security 54
- T-DES (Triple DES) 37
- technologies 31, 45
- Technology or application weakness 18
- The local policy replica data store 109
- Threat analysis 9
- threat assessment 196
- threat modelling 17
- threats 16, 196
- Time Stamp 6
- Time Stamp Authority (TSA) 7
- Top CA 7
- Transaction-level security 47
- Triple DES 34, 37, 185
- types of attacks 133

U

- UNIX security 52
- User ID and reslevel 80
- user identity mapping 124
- User Registry 117
- user-written exits 95

V

- Verify SSL client 244
- verify the partner connection 105
- Virtual Private Networks (VPN) 47
- vulnerabilities 1, 186
- vulnerability assessment 196

W

- waterfall model 11
- WebSphere MQ AP 100
- WebSphere MQ data flow 100
- WebSphere MQ denial of service attack 187
- WebSphere MQ Explorer GUI 165
- WebSphere MQ objects for SSL 40
- WebSphere MQ security 167
- WebSphere MQ setup 238
- WebSphere MQ setup to implement SSL 205
- WebSphere MQ technologies 5
- WebSphere MQ trigger monitor (CKTI) 178
- When to use AMBI 127
- Windows 2000 security 57
- Wireless lans 190
- WITHLABE 224

X

- xSeries 5, 148

Z

- z/OS security 46
- z/OS Security Server 48
- z/OS System Authorization Facility (SAF) 63
- ZPARM 68
- zSeries 5, 146



WebSphere MQ Security in an Enterprise Environment

(0.5" spine)
0.475" <-> 0.875"
250 <-> 459 pages



Redbooks

WebSphere MQ Security in an Enterprise Environment

Cross-platform security

Secure Sockets Layer

Message security

This IBM Redbook considers an enterprise and describes some of the procedures and documentation that need to be developed to secure WebSphere MQ on the z/OS (zSeries), OS/400 (iSeries), IBM AIX (pSeries) and Windows 2000 (xSeries) platforms. This redbook also documents the before and after configurations needed to take advantage of the recent functional improvements to WebSphere MQ, such as the Secure Sockets Layer (SSL).

Security is a complex subject. The first part of this redbook is intended to help the reader understand it. The second part lays out a business case scenario where the technology is implemented to secure WebSphere MQ.

The book incorporates:

- Cross-platform security, ascertaining the responsibility of identification and authentication.
- Security issues when using WebSphere MQ over the Internet.
- Securing of messages, taking advantage of the functional improvements to WebSphere MQ such as PKI and SSL.
- Maintenance of message integrity by means of authentication and encryption.
- Use of technologies external to WebSphere MQ, such as PKI, DCE, and RACF to solve the security concerns.

The appendixes provide additional information, scripts, sample code and JCL related to the chapters in the book.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6814-00

ISBN 0738425621