System Administration Guide

Contents

1. Introduction
   1.1. Introduction to WebSphere MQ
      1.1.1. WebSphere MQ and message queuing
         1.1.1.1. Time-independent applications
         1.1.1.2. Message-driven processing
      1.1.2. Messages and queues
         1.1.2.1. What is a message?
            1.1.2.1.1. Message lengths
            1.1.2.1.2. How do applications send and receive messages?
         1.1.2.2. What is a queue?
            1.1.2.2.1. Predefined queues and dynamic queues
            1.1.2.2.2. Retrieving messages from queues
      1.1.3. Objects
         1.1.3.1. Object names
         1.1.3.2. Managing objects
         1.1.3.3. Object attributes
         1.1.3.4. WebSphere MQ queues
            1.1.3.4.1. Defining queues
            1.1.3.4.2. Queues used by WebSphere MQ
         1.1.3.5. WebSphere MQ queue managers
         1.1.3.6. Process definitions
         1.1.3.7. Clusters
         1.1.3.8. Namelists
         1.1.3.9. Authentication information objects
         1.1.3.10. Channels
         1.1.3.11. Client connection channels
         1.1.3.12. Listeners
         1.1.3.13. Services
         1.1.3.14. Administrative topic objects
         1.1.3.15. System default objects
      1.1.4. Clients and servers
         1.1.4.1. WebSphere MQ applications in a client-server environment
      1.1.5. Extending queue manager facilities
         1.1.5.1. User exits
         1.1.5.2. API exits
         1.1.5.3. Installable services
      1.1.6. Security
         1.1.6.1. Object Authority Manager (OAM) facility
         1.1.6.2. User-written or third party channel exits
         1.1.6.3. Channel security using SSL
      1.1.7. Transactional support
   1.2. An introduction to WebSphere MQ administration
      1.2.1. Local and remote administration
      1.2.2. Performing administration tasks using commands
         1.2.2.1. Control commands
         1.2.2.2. WebSphere MQ Script (MQSC) commands
         1.2.2.3. PCF commands
      1.2.3. Further methods of administration
         1.2.3.1. Using the WebSphere MQ Explorer
         1.2.3.2. Using the Windows Default Configuration application
         1.2.3.3. Using the Microsoft Cluster Service (MSCS)
      1.2.4. Understanding WebSphere MQ file names
         1.2.4.1. Queue manager name transformation
         1.2.4.2. Object name transformation
2. Administration using WebSphere MQ commands
   2.1. Managing queue managers
      2.1.1. Using control commands
         2.1.1.1. Using control commands on Windows systems
         2.1.1.2. Using control commands on UNIX systems
      2.1.2. Using the WebSphere MQ Explorer
      2.1.3. Creating a queue manager
         2.1.3.1. Guidelines for creating queue managers
         2.1.3.2. Creating a default queue manager
         2.1.3.3. Making an existing queue manager the default
            2.1.3.3.1. Windows and WebSphere MQ for Linux (x86 platform) systems
            2.1.3.3.2. UNIX systems
         2.1.3.4. Backing up configuration files after creating a queue manager
            2.1.3.4.1. Windows and WebSphere MQ for Linux (x86 platform) systems
            2.1.3.4.2. UNIX systems
      2.1.4. Starting a queue manager
         2.1.4.1. Starting a queue manager automatically
      2.1.5. Stopping a queue manager
         2.1.5.1. Quiesced shutdown
         2.1.5.2. Immediate shutdown
         2.1.5.3. Preemptive shutdown
         2.1.5.4. If you have problems shutting down a queue manager
      2.1.6. Restarting a queue manager
      2.1.7. Deleting a queue manager
   2.2. Administering local WebSphere MQ objects
      2.2.1. Supporting application programs that use the MQI
      2.2.2. Performing local administration tasks using MQSC commands
         2.2.2.1. WebSphere MQ object names
            2.2.2.1.1. Case-sensitivity in MQSC commands
         2.2.2.2. Standard input and output
         2.2.2.3. Using MQSC commands interactively
            2.2.2.3.1. Feedback from MQSC commands
            2.2.2.3.2. Ending interactive input of MQSC commands
         2.2.2.4. Running MQSC commands from text files
            2.2.2.4.1. MQSC command files
            2.2.2.4.2. MQSC command reports
            2.2.2.4.3. Running the supplied MQSC command files
            2.2.2.4.4. Using runmqsc to verify commands
         2.2.2.5. Running MQSC commands from batch files

2.2.2.6. Resolving problems with MQSC commands
2.2.3. Working with queue managers
2.2.3.1. Displaying queue manager attributes
2.2.3.2. Altering queue manager attributes
2.2.4. Working with local queues
2.2.4.1. Defining a local queue
2.2.4.1.1. Defining a dead-letter queue
2.2.4.2. Displaying default object attributes
2.2.4.3. Copying a local queue definition
2.2.4.4. Changing local queue attributes
2.2.4.5. Clearing a local queue
2.2.4.6. Deleting a local queue
2.2.4.7. Browsing queues
2.2.4.8. Monitoring local queues with the Windows Performance Monitor
2.2.4.9. Enabling large queues
2.2.5. Working with alias queues
2.2.5.1. Defining an alias queue
2.2.5.2. Using other commands with alias queues
2.2.6. Working with model queues
2.2.6.1. Defining a model queue
2.2.6.2. Using other commands with model queues
2.2.7. Working with services
2.2.7.1. Defining a service object
2.2.7.2. Managing services
2.2.7.3. Additional environment variables
2.2.7.4. Replaceable inserts on service definitions
2.2.7.4.1. Common tokens
2.2.7.5. Examples on using service objects
2.2.7.5.1. Using a server service object
2.2.7.5.2. Using a command service object
2.2.7.5.3. Using a command service object when a queue manager ends only
2.2.7.5.4. More on passing arguments
2.2.7.5.5. Autostarting a Service
2.2.8. Managing objects for triggering
2.2.8.1. Defining an application queue for triggering
2.2.8.2. Defining an initiation queue
2.2.8.3. Defining a process
2.2.8.4. Displaying attributes of a process definition
2.3. Automating administration tasks
2.3.1. PCF commands
2.3.1.1. PCF object attributes
2.3.1.2. Escape PCFs
2.3.1.3. Using the MQAI to simplify the use of PCFs
2.3.1.4. Active Directory Services Interfaces
2.3.1.4.1. Client connection channels in the Active Directory
2.4. Administering remote WebSphere MQ objects
2.4.1. Channels, clusters, and remote queuing
2.4.1.1. Remote administration using clusters
2.4.2. Remote administration from a local queue manager
2.4.2.1. Preparing queue managers for remote administration
2.4.2.2. Preparing channels and transmission queues for remote administration
2.4.2.2.1. Defining channels, listeners, and transmission queues
2.4.2.2.2. Starting the listeners and channels
2.4.2.2.2.1. Automatic definition of channels
2.4.2.3. Managing the command server for remote administration
2.4.2.3.1. Starting the command server
2.4.2.3.2. Displaying the status of the command server
2.4.2.3.3. Stopping a command server
2.4.2.4. Issuing MQSC commands on a remote queue manager
2.4.2.4.1. Working with queue managers on z/OS
2.4.2.5. Recommendations for issuing commands remotely
2.4.2.6. If you have problems using MQSC commands remotely
2.4.3. Creating a local definition of a remote queue
2.4.3.1. Understanding how local definitions of remote queues work
2.4.3.1.1. Example
2.4.3.1.1.1. Purpose
2.4.3.1.1.2. How it works
2.4.3.2. An alternative way of putting messages on a remote queue
2.4.3.3. Using other commands with remote queues
2.4.3.4. Defining a transmission queue
2.4.3.4.1. Default transmission queues
2.4.4. Using remote queue definitions as aliases
2.4.4.1. Queue manager aliases
2.4.4.2. Reply-to queue aliases
2.4.5. Data conversion
2.4.5.1. When a queue manager cannot convert messages in built-in formats
2.4.5.2. File ccsid.tbl
2.4.5.2.1. Default data conversion
2.4.5.3. Converting messages in user-defined formats
2.4.5.4. Changing the queue manager CCSID
3. Administration using the WebSphere MQ Explorer
3.1. What you can do with the WebSphere MQ Explorer
3.1.1. Remote queue managers
3.1.2. Deciding whether to use the WebSphere MQ Explorer
3.2. Setting up the WebSphere MQ Explorer
3.2.1. Prerequisite software
3.2.2. Required definitions for administration
3.2.3. Cluster membership
3.2.4. Security
3.2.4.1. Authorization to use the WebSphere MQ Explorer
3.2.4.2. Security for connecting to remote queue managers
3.2.4.3. Using a security exit
3.2.4.4. Using SSL security
3.2.4.5. Using the WebSphere MQ Explorer to connect to a remote queue manager using SSL enabled MQI channels
3.2.4.6. Connecting through another queue manager
3.2.5. Data conversion
3.3. Using the WebSphere MQ Explorer
3.3.1. Showing and hiding queue managers and clusters

3.3.2. Starting the WebSphere MQ Explorer in standalone mode or Eclipse workbench mode
3.4. Security on Windows
3.4.1. Using Active directory (Windows only)
3.4.1.1. User rights required for AMQMSRVN
3.4.1.2. Changing the user name associated with WebSphere MQ Services
3.4.2. Controlling access (Windows only)
3.4.2.1. Using DCOMCNFG.EXE to change access permissions
3.4.3. Changing the password of the AMQMSRVN user account
3.4.4. WebSphere MQ coordinating with DB2 as the resource manager
3.5. Extending the WebSphere MQ Explorer
3.5.1. Who this chapter is for
3.5.2. What you need to know to understand this chapter
3.5.3. Introduction
3.5.3.1. Importing the sample Eclipse plug-ins
3.5.4. Writing an Eclipse plug-in for the WebSphere MQ Explorer
3.5.4.1. Accessing Javadoc
3.5.4.2. Utilizing extension points
3.5.4.2.1. Register
3.5.4.2.1.1. Enabling and disabling a plug-in
3.5.4.2.1.2. Notify events
3.5.4.2.2. Add tree node
3.5.4.2.3. Add content page
3.5.4.2.4. Add context menu item
3.5.4.2.5. Add property tab
3.5.5. Applying plug-ins to the WebSphere MQ Explorer
4. Using the WebSphere MQ Taskbar application (Windows only)
4.1. Using the WebSphere MQ alert monitor application (Windows only)
5. WebSphere MQ control commands
5.1. How to use WebSphere MQ control commands
5.1.1. Names of WebSphere MQ objects
5.1.2. Syntax diagrams
5.1.2.1. How to read railroad diagrams
5.1.2.2. How to read dotted decimal diagrams
5.1.3. Example syntax diagram
5.1.4. Syntax help
5.1.4.1. Examples
5.2. The control commands
5.2.1. addmqinf (Add configuration information)
5.2.2. amqccert (check certificate chains)
5.2.3. amqmdain (WebSphere MQ services control)
5.2.4. amqmfsck (file system check)
5.2.5. amqtcert (transfer certificates)
5.2.5.1. Listing the contents of certificate stores
5.2.5.2. Manually migrating certificate stores
5.2.5.3. Automatically migrating certificate stores
5.2.5.4. Listing the contents of registry entries
5.2.5.5. Removing state information
5.2.6. crtmqcvx (data conversion)
5.2.7. crtmqm (create queue manager)
5.2.8. dltmqm (delete queue manager)
5.2.9. dmpmqaut (dump authority)
5.2.10. dmpmqlog (dump log)
5.2.11. dspmq (display queue managers)
5.2.12. dspmqaut (display authority)
5.2.13. dspmqcsv (display command server)
5.2.14. dspmqfls (display files)
5.2.15. dspmqinf (Display configuration information)
5.2.16. dspmqrte (WebSphere MQ display route application)
5.2.17. dspmqtrc (display formatted trace output)
5.2.18. dspmqtrn (display transactions)
5.2.19. dspmqver (display version information)
5.2.20. endmqcsv (end command server)
5.2.21. endmqlsr (end listener)
5.2.22. endmqdnm (stop .NET monitor)
5.2.23. endmqm (end queue manager)
5.2.24. endmqtrc (end trace)
5.2.25. migmbbrk (migrate publish/subscribe information)
5.2.26. mqftapp (run File Transfer Application GUI)
5.2.27. mqftrcv (receive file on server)
5.2.28. mqftrcvc (receive file on client)
5.2.29. mqftsnd (send file from server)
5.2.30. mqftsndc (send file from client)
5.2.31. rcdmqimg (record media image)
5.2.32. rcrmqobj (recreate object)
5.2.33. rmvmqinf (Remove configuration information)
5.2.34. rsvmqtrn (resolve transactions)
5.2.35. runmqchi (run channel initiator)
5.2.36. runmqchl (run channel)
5.2.37. runmqdlq (run dead-letter queue handler)
5.2.38. runmqdnm (run .NET monitor)
5.2.39. runmqlsr (run listener)
5.2.40. runmqsc (run MQSC commands)
5.2.41. runmqtmc (start client trigger monitor)
5.2.42. runmqtrm (start trigger monitor)
5.2.43. setmqaut (grant or revoke authority)
5.2.43.1. Authorizations for MQI calls
5.2.43.2. Authorizations for context
5.2.43.3. Authorizations for commands
5.2.43.4. Authorizations for generic operations
5.2.44. setmqcrl (set certificate revocation list (CRL) LDAP server definitions)
5.2.45. setmqprd (enroll production license)
5.2.46. setmqscp (set service connection points)
5.2.47. strmqcfg (start WebSphere MQ Explorer)
5.2.48. strmqcsv (start command server)
5.2.49. strmqm (start queue manager)
5.2.50. strmqtrc (Start trace)
5.3. Comparing command sets
5.3.1. Queue manager commands

5.3.2. Command server commands
5.3.3. Authority commands
5.3.4. Cluster commands
5.3.5. Authentication information commands
5.3.6. Channel commands
5.3.7. Listener commands
5.3.8. Namelist commands
5.3.9. Process commands
5.3.10. Queue commands
5.3.11. Service commands
5.3.12. Other commands
5.4. Managing keys and certificates
5.4.1. Preparing to use the gsk7cmd and gsk7capicmd commands
5.4.2. gsk7cmd, runmqckm, and gsk7capicmd commands
5.4.2.1. Commands for a CMS key database only
5.4.2.2. Commands for CMS or PKCS #12 key databases
5.4.2.3. Commands for cryptographic device operations
5.4.3. gsk7cmd, runmqckm, and gsk7capicmd options
5.4.4. gsk7capicmd error codes
6. Configuration and management
6.1. Configuring WebSphere MQ
6.1.1. Changing configuration information on Windows systems
6.1.1.1. Viewing configuration information
6.1.2. Changing configuration information on UNIX systems
6.1.2.1. Editing configuration files
6.1.2.1.1. When do you need to edit a configuration file?
6.1.2.1.2. Configuration file priorities
6.1.2.2. The WebSphere MQ configuration file, mqs.ini
6.1.2.3. Queue manager configuration files, qm.ini
6.1.3. Attributes for changing WebSphere MQ configuration information
6.1.3.1. All queue managers
6.1.3.2. Default queue manager
6.1.3.3. Exit properties
6.1.3.4. Log defaults for WebSphere MQ
6.1.3.5. Advanced Configuration and Power Interface (ACPI)
6.1.3.6. API exits
6.1.3.7. Queue managers
6.1.4. Changing queue manager configuration information
6.1.4.1. Installable services
6.1.4.1.1. Service components
6.1.4.2. Queue manager logs
6.1.4.3. Restricted mode
6.1.4.4. XA resource managers
6.1.4.5. Channels
6.1.4.6. LU62, NETBIOS, TCP, and SPX
6.1.4.7. Exit path
6.1.4.7.1. API exits
6.1.4.8. Queue manager error logs
6.1.4.9. Queue manager default bind type
6.1.4.10. SSL stanza of the queue manager configuration file
6.1.4.11. Exit properties
6.1.4.12. Subpool
6.1.5. System and default objects
6.1.5.1. Windows default configuration objects
6.1.5.2. SYSTEM.BASE.TOPIC
6.2. File system
6.2.1. Requirements for shared file systems
6.2.1.1. Verifying shared file system locking
6.2.2. Sharing WebSphere MQ files
6.2.3. Directory structure on UNIX and Linux systems
6.2.3.1. Directory content on UNIX and Linux systems
6.2.3.2. Example directory configurations on Unix systems
6.2.4. Directory structure on Windows
6.2.4.1. Directory content
6.3. WebSphere MQ security
6.3.1. Common Criteria
6.3.2. Authority to administer WebSphere MQ
6.3.2.1. Managing the mqm group
6.3.3. Authority to work with WebSphere MQ objects
6.3.3.1. When security checks are made
6.3.3.2. How access control is implemented by WebSphere MQ
6.3.3.3. Identifying the user ID
6.3.3.3.1. Principals and groups
6.3.3.3.2. Windows security identifiers (SIDs)
6.3.3.4. Alternate-user authority
6.3.3.5. Context authority
6.3.4. Connecting to WebSphere MQ using Terminal Services
6.3.5. Configuring additional authority for Windows applications connecting to WebSphere MQ
6.3.6. Creating and managing groups
6.3.6.1. Windows
6.3.6.1.1. Creating a group
6.3.6.1.2. Adding a user to a group
6.3.6.1.3. Displaying who is in a group
6.3.6.1.4. Removing a user from a group
6.3.6.2. HP-UX
6.3.6.2.1. Creating a group
6.3.6.2.2. Adding a user to a group
6.3.6.2.3. Displaying who is in a group
6.3.6.2.4. Removing a user from a group
6.3.6.3. AIX
6.3.6.3.1. Creating a group
6.3.6.3.2. Adding a user to a group
6.3.6.3.3. Displaying who is in a group
6.3.6.3.4. Removing a user from a group
6.3.6.4. Solaris
6.3.6.4.1. Creating a group
6.3.6.4.2. Adding a user to a group
6.3.6.4.3. Displaying who is in a group

6.3.6.4.4. Removing a user from a group
6.3.6.5. Linux
6.3.6.5.1. Creating a group
6.3.6.5.2. Adding a user to a group
6.3.6.5.3. Displaying who is in a group
6.3.6.5.4. Removing a user from a group
6.3.7. Using the OAM to control access to objects
6.3.7.1. Giving access to a WebSphere MQ object
6.3.7.1.1. Examples of using the command
6.3.7.1.2. Using the command with a different authorization service
6.3.7.2. Using OAM generic profiles
6.3.7.2.1. Using wildcard characters
6.3.7.2.2. Profile priorities
6.3.7.2.3. Dumping profile settings
6.3.7.3. Displaying access settings
6.3.7.4. Changing and revoking access to a WebSphere MQ object
6.3.7.5. Preventing security access checks
6.3.8. Channel security
6.3.8.1. Protecting channel initiator definitions
6.3.8.2. Transmission queues
6.3.8.3. Channel exits
6.3.8.4. Protecting channels with SSL
6.3.9. How authorizations work
6.3.9.1. Authorizations for MQI calls
6.3.9.2. Authorizations for MQSC commands in escape PCFs
6.3.9.3. Authorizations for PCF commands
6.3.10. Special considerations for security on Windows
6.3.10.1. When you get a 'group not found' error
6.3.10.2. When you have problems with WebSphere MQ and domain controllers
6.3.10.2.1. Windows 2000 domain with non-default, or Windows 2003 domain with default, security permissions
6.3.10.2.2. Configuring WebSphere MQ Services to run under a domain user
6.3.10.3. Applying security template files
6.3.10.4. Nested groups
6.4. Transactional support
6.4.1. Introducing units of work
6.4.2. Scenario 1: Queue manager performs the coordination
6.4.2.1. Database coordination
6.4.2.1.1. Restrictions
6.4.2.1.2. Switch load files
6.4.2.1.3. Configuring your system for database coordination
6.4.2.1.3.1. Installing and configuring the database product
6.4.2.1.3.1.1. Database connections
6.4.2.1.3.1.2. Client/server configuration
6.4.2.1.3.2. Creating switch load files
6.4.2.1.3.3. Adding configuration information to the queue manager
6.4.2.1.3.4. Writing and modifying your applications
6.4.2.1.3.5. Testing the system
6.4.2.2. DB2 configuration
6.4.2.2.1. Checking the DB2 environment variable settings
6.4.2.2.2. Creating the DB2 switch load file
6.4.2.2.3. Adding resource manager configuration information for DB2
6.4.2.2.4. Changing DB2 configuration parameters
6.4.2.3. Oracle configuration
6.4.2.3.1. Checking the Oracle environment variable settings
6.4.2.3.2. Creating the Oracle switch load file
6.4.2.3.3. Adding resource manager configuration information for Oracle
6.4.2.3.4. Changing Oracle configuration parameters
6.4.2.4. Informix configuration
6.4.2.4.1. Ensuring Informix databases are created correctly
6.4.2.4.2. Checking the Informix environment variable settings
6.4.2.4.3. Creating the Informix switch load file
6.4.2.4.4. Adding resource manager configuration information for Informix
6.4.2.5. Sybase configuration
6.4.2.5.1. Checking the Sybase environment variable settings
6.4.2.5.2. Enabling Sybase XA support
6.4.2.5.3. Creating the Sybase switch load file
6.4.2.5.4. Adding resource manager configuration information for Sybase
6.4.2.5.5. Using multi-threaded programs with Sybase
6.4.2.6. Multiple database configurations
6.4.2.7. Security considerations
6.4.2.8. Administration tasks
6.4.2.8.1. In-doubt units of work
6.4.2.8.2. Displaying outstanding units of work with the dspmqtrn command
6.4.2.8.3. Resolving outstanding units of work with the rsvmqtrn command
6.4.2.8.4. Mixed outcomes and errors
6.4.2.8.5. Changing configuration information
6.4.2.8.5.1. Removing database manager instances
6.4.2.9. XA dynamic registration
6.4.2.9.1. Error conditions
6.4.2.9.2. Summarizing XA calls
6.4.3. Scenario 2: Other software provides the coordination
6.4.3.1. External sync point coordination
6.4.3.1.1. The WebSphere MQ XA switch structure
6.4.3.2. Using CICS
6.4.3.2.1. The CICS two-phase commit process
6.4.3.2.1.1. Requirements of the two-phase commit process
6.4.3.2.1.2. Switch load files
6.4.3.2.1.2.1. Building libraries for use with TXSeries® for Multiplatforms version 5
6.4.3.2.1.2.2. Building libraries for use with TXSeries® for Multiplatforms version 6
6.4.3.2.1.2.3. WebSphere MQ XA support and Tuxedo
6.4.3.2.1.3. Enabling the CICS two-phase commit process
6.4.3.2.1.4. Enabling CICS user exits
6.4.3.2.1.4.1. Sample task termination exit
6.4.3.3. Using the Microsoft Transaction Server (COM+)
6.4.4. Unit of recovery disposition
6.5. The WebSphere MQ dead-letter queue handler
6.5.1. Invoking the DLQ handler
6.5.1.1. The sample DLQ handler, amqsdlq

6.5.2. The DLQ handler rules table
6.5.2.1. Control data
6.5.2.2. Rules (patterns and actions)
6.5.2.2.1. The pattern-matching keywords
6.5.2.2.2. The action keywords
6.5.2.3. Rules table conventions
6.5.3. How the rules table is processed
6.5.3.1. Ensuring that all DLQ messages are processed
6.5.4. An example DLQ handler rules table
6.6. Availability, recovery and restart
6.6.1. Automatic client reconnection
6.6.2. Stopping and removing queue managers manually
6.6.2.1. Stopping a queue manager manually
6.6.2.1.1. Stopping queue managers in WebSphere MQ for Windows
6.6.2.1.2. Stopping queue managers in WebSphere MQ for UNIX systems
6.6.2.2. Removing queue managers manually
6.6.2.2.1. Removing queue managers in WebSphere MQ for Windows
6.6.2.2.1.1. Removing queue managers from the automatic startup list
6.6.2.2.2. Removing queue managers in WebSphere MQ for UNIX systems
6.6.3. High availability configurations
6.6.3.1. HA clusters on UNIX
6.6.3.1.1. Configuring the shared disks
6.6.3.1.2. Creating a queue manager on a node
6.6.3.1.3. Adding queue manager configuration information to other nodes
6.6.3.1.4. Starting a queue manager
6.6.3.1.5. Stopping a queue manager
6.6.3.1.6. Monitoring a queue manager
6.6.3.1.7. Putting the queue manager under control of the HA cluster
6.6.3.1.8. Deleting a queue manager
6.6.3.2. HA cluster on Windows - Microsoft Cluster Service (MSCS)
6.6.3.2.1. Introducing MSCS clusters
6.6.3.2.2. Setting up WebSphere MQ for MSCS clustering
6.6.3.2.2.1. Setup symmetry
6.6.3.2.2.2. MSCS security
6.6.3.2.2.3. Using multiple queue managers with MSCS
6.6.3.2.2.4. Cluster modes
6.6.3.2.2.4.1. Active/Passive mode
6.6.3.2.2.4.2. Active/Active mode
6.6.3.2.3. Creating a queue manager for use with MSCS
6.6.3.2.3.1. Creating a queue manager from a command prompt
6.6.3.2.3.2. Creating a queue manager using the WebSphere MQ Explorer
6.6.3.2.4. Moving a queue manager to MSCS storage
6.6.3.2.5. Putting a queue manager under MSCS control
6.6.3.2.6. Summary of looksAlive and isAlive polling intervals
6.6.3.2.7. Removing a queue manager from MSCS control
6.6.3.2.7.1. Taking a queue manager offline from MSCS
6.6.3.2.7.2. Returning a queue manager from MSCS storage
6.6.3.2.8. Hints and tips on using MSCS
6.6.3.2.8.1. Verifying that MSCS is working
6.6.3.2.8.2. Manual startup
6.6.3.2.8.3. MSCS and queue managers
6.6.3.2.8.3.1. Creating a matching queue manager on the other node
6.6.3.2.8.3.2. Default queue managers
6.6.3.2.8.3.3. Deleting a queue manager
6.6.3.2.8.3.4. Support for existing queue managers
6.6.3.2.8.3.5. Telling MSCS which queue managers to manage
6.6.3.2.8.3.6. Queue manager log files
6.6.3.2.8.3.7. Multiple queue managers
6.6.3.2.8.4. Always use MSCS to manage clusters
6.6.3.2.8.5. Working in Active/Active mode
6.6.3.2.8.6. PostOnlineCommand and PreOfflineCommand
6.6.3.2.8.7. Using preferred nodes
6.6.3.2.8.8. If COM+ errors occur in the Application Event log
6.6.3.2.9. WebSphere MQ MSCS support utility programs
6.6.3.3. Multi-instance queue managers
6.6.3.3.1. Creating a multi-instance queue manager
6.6.3.3.1.1. Create a multi-instance queue manager on Windows Server
6.6.3.3.1.1.1. Windows domains and multi-instance queue managers
6.6.3.3.1.1.2. Verify the multi-instance queue manager on Windows
6.6.3.3.1.2. Create a multi-instance queue manager on Linux
6.6.3.3.1.2.1. Verify the multi-instance queue manager on Linux
6.6.3.3.2. Deleting a multi-instance queue manager
6.6.3.3.3. Starting and stopping a multi-instance queue manager
6.6.3.3.4. Shared file system
6.6.3.3.5. Multiple queue manager instances
6.6.3.3.6. Failover or switchover
6.6.3.3.7. Channel and client reconnection
6.6.3.3.8. Application recovery
6.6.3.3.9. Data recovery and high availability
6.6.3.3.10. Combining MQ Availability solutions
6.6.4. Making sure that messages are not lost (logging)
6.6.4.1. What logs look like
6.6.4.1.1. The log control file
6.6.4.2. Types of logging
6.6.4.2.1. Circular logging
6.6.4.2.2. Linear logging
6.6.4.3. Using checkpointing to ensure complete recovery
6.6.4.3.1. Checkpointing with long-running transactions
6.6.4.4. Calculating the size of the log
6.6.4.5. Managing logs
6.6.4.5.1. What happens when a disk gets full
6.6.4.5.2. Managing log files
6.6.4.5.2.1. Determining superfluous log files
6.6.4.5.2.2. Log file location
6.6.4.6. Using the log for recovery
6.6.4.6.1. Recovering from power loss or communications failures
6.6.4.6.2. Recovering damaged objects
6.6.4.6.2.1. Media recovery

6.6.4.6.2.2. Recovering from media images
6.6.4.6.2.3. Recovering damaged objects during start up
6.6.4.6.2.4. Recovering damaged objects at other times
6.6.4.7. Protecting WebSphere MQ log files
6.6.4.8. Dumping the contents of the log using the dmpmqlog command
6.6.5. Backing up and restoring WebSphere MQ
6.6.5.1. Backing up queue manager data
6.6.5.2. Restoring queue manager data
6.6.5.3. Using a backup queue manager
6.6.5.4. Creating a backup queue manager
6.6.5.5. Updating a backup queue manager
6.6.5.6. Starting a backup queue manager
6.6.6. Recovery scenarios
6.6.6.1. Disk drive failures
6.6.6.2. Damaged queue manager object
6.6.6.3. Damaged single object
6.6.6.4. Automatic media recovery failure
6.7. Problem determination
6.7.1. Preliminary checks
6.7.1.1. Has WebSphere MQ run successfully before?
6.7.1.2. Are there any error messages?
6.7.1.3. Does the Windows Application Event Log show any WebSphere MQ errors? (Windows only)
6.7.1.4. Are there any return codes explaining the problem?
6.7.1.5. Can you reproduce the problem?
6.7.1.6. Have any changes been made since the last successful run?
6.7.1.7. Has the application run successfully before?
6.7.1.7.1. If the application has not run successfully before
6.7.1.7.2. Common programming errors
6.7.1.8. Problems with commands
6.7.1.9. Does the problem affect specific parts of the network?
6.7.1.10. Does the problem occur at specific times of the day?
6.7.1.11. Is the problem intermittent?
6.7.1.12. Have you applied any service updates?
6.7.2. Looking at problems in more detail
6.7.2.1. Have you obtained incorrect output?
6.7.2.1.1. Messages that do not appear on the queue
6.7.2.1.2. Messages that contain unexpected or corrupted information
6.7.2.1.3. Problems with incorrect output when using distributed queues
6.7.2.2. Have you failed to receive a response from a PCF command?
6.7.2.3. Are some of your queues failing?
6.7.2.4. Are you receiving an error code when creating or starting a queue manager? (Windows only)
6.7.2.5. Does the problem affect only remote queues?
6.7.2.6. Is your application or system running slowly?
6.7.2.6.1. Tuning performance for nonpersistent messages on AIX
6.7.3. Application design considerations
6.7.3.1. Effect of message length
6.7.3.2. Effect of message persistence
6.7.3.3. Searching for a particular message
6.7.3.4. Queues that contain messages of different lengths
6.7.3.5. Frequency of sync points
6.7.3.6. Use of the MQPUT1 call
6.7.3.7. Number of threads in use
6.7.4. Error logs
6.7.4.1. Error log files
6.7.4.1.1. Early errors
6.7.4.1.2. An example of an error log
6.7.4.2. Error log access restrictions under UNIX systems
6.7.4.3. Ignoring error codes under UNIX systems
6.7.4.4. Ignoring error codes under Windows systems
6.7.4.5. Operator messages
6.7.5. Dead-letter queues
6.7.6. Configuration files and problem determination
6.7.7. Tracing
6.7.7.1. Tracing WebSphere MQ for Windows
6.7.7.1.1. Selective component tracing on WebSphere MQ for Windows
6.7.7.1.2. Selective process tracing on WebSphere MQ for Windows
6.7.7.1.3. Trace files
6.7.7.1.4. An example of WebSphere MQ for Windows trace data
6.7.7.2. Tracing WebSphere MQ for UNIX systems
6.7.7.2.1. Selective component tracing on WebSphere MQ for UNIX systems
6.7.7.2.2. Selective process tracing on WebSphere MQ for UNIX systems
6.7.7.2.3. Example trace data for WebSphere MQ for UNIX systems
6.7.7.3. Trace files
6.7.7.4. Tracing Secure Sockets Layer (SSL) iKeyman and iKeycmd functions
6.7.7.5. Tracing with the AIX system trace
6.7.7.5.1. Selective component tracing on WebSphere MQ for AIX
6.7.7.5.1.1. SSL trace
6.7.7.5.2. An example of WebSphere MQ for AIX trace data
6.7.8. First-failure support technology (FFST)
6.7.8.1. FFST: WebSphere MQ for Windows
6.7.8.2. FFST: WebSphere MQ for UNIX systems
6.7.9. Problem determination with WebSphere MQ clients
6.7.9.1. Terminating clients
6.7.10. Java diagnostics
6.7.10.1. Using com.ibm.mq.commonservices
6.7.10.2. Java trace and FFDC files
6.8. WebSphere MQ and UNIX System V IPC resources
6.8.1. Clearing WebSphere MQ shared memory resources
6.8.2. Shared memory on AIX
6.9. WebSphere MQ and UNIX Process Priority
7. WebSphere MQ installable services and the API exit
7.1. Installable services and components
7.1.1. Why installable services?
7.1.2. Functions and components
7.1.2.1. Entry-points
7.1.2.2. Return codes
7.1.2.3. Component data
7.1.3. Initialization

7.1.3.1. Primary initialization
7.1.3.2. Secondary initialization
7.1.3.3. Primary termination
7.1.3.4. Secondary termination
7.1.4. Configuring services and components
7.1.4.1. Service stanza format
7.1.4.2. Service stanza format for Windows systems
7.1.4.3. Service component stanza format
7.1.5. Creating your own service component
7.1.6. Using multiple service components
7.1.6.1. Example of using multiple components
7.1.6.1.1. What the components do
7.1.6.1.2. How the component is used
7.1.6.2. Omitting entry points when using multiple components
7.1.6.3. Example of entry points used with multiple components
7.2. Authorization service
7.2.1. Object authority manager (OAM)
7.2.1.1. Defining the service to the operating system
7.2.1.2. Refreshing the OAM after changing a user's authorization
7.2.1.3. Migrating from MQSeries Version 5.1
7.2.2. Authorization service on UNIX systems
7.2.2.1. Configuring authorization service stanzas: UNIX systems
7.2.3. Authorization service on Windows systems
7.2.3.1. Configuring authorization service stanzas: Windows systems
7.2.4. Authorization service interface
7.3. Name service
7.3.1. How the name service works
7.3.1.1. Name service interface
7.4. Installable services interface reference information
7.4.1. How the functions are shown
7.4.2. MQZEP – Add component entry point
7.4.2.1. Hconfig (MQHCONFIG) – input
7.4.2.2. Function (MQLONG) – input
7.4.2.3. EntryPoint (PMQFUNC) – input
7.4.2.4. CompCode (MQLONG) – output
7.4.2.5. Reason (MQLONG) – output
7.4.3. MQZ_AUTHENTICATE_USER – Authenticate user
7.4.3.1. QMgrName (MQCHAR48) – input
7.4.3.2. SecurityParms (MQCSP) – input
7.4.3.3. ApplicationContext (MQZAC) – input
7.4.3.4. IdentityContext (MQZIC) – input/output
7.4.3.5. CorrelationPtr (MQPTR) – output
7.4.3.6. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.3.7. Continuation (MQLONG) – output
7.4.3.8. CompCode (MQLONG) – output
7.4.3.9. Reason (MQLONG) – output
7.4.4. MQZ_CHECK_AUTHORITY – Check authority
7.4.4.1. QMgrName (MQCHAR48) – input
7.4.4.2. EntityName (MQCHAR12) – input
7.4.4.3. EntityType (MQLONG) – input
7.4.4.4. ObjectName (MQCHAR48) – input
7.4.4.5. ObjectType (MQLONG) – input
7.4.4.6. Authority (MQLONG) – input
7.4.4.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.4.8. Continuation (MQLONG) – output
7.4.4.9. CompCode (MQLONG) – output
7.4.4.10. Reason (MQLONG) – output
7.4.5. MQZ_CHECK_AUTHORITY_2 – Check authority (extended)
7.4.5.1. QMgrName (MQCHAR48) – input
7.4.5.2. EntityData (MQZED) – input
7.4.5.3. EntityType (MQLONG) – input
7.4.5.4. ObjectName (MQCHAR48) – input
7.4.5.5. ObjectType (MQLONG) – input
7.4.5.6. Authority (MQLONG) – input
7.4.5.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.5.8. Continuation (MQLONG) – output
7.4.5.9. CompCode (MQLONG) – output
7.4.5.10. Reason (MQLONG) – output
7.4.6. MQZ_COPY_ALL_AUTHORITY – Copy all authority
7.4.6.1. QMgrName (MQCHAR48) – input
7.4.6.2. RefObjectName (MQCHAR48) – input
7.4.6.3. ObjectName (MQCHAR48) – input
7.4.6.4. ObjectType (MQLONG) – input
7.4.6.5. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.6.6. Continuation (MQLONG) – output
7.4.6.7. CompCode (MQLONG) – output
7.4.6.8. Reason (MQLONG) – output
7.4.7. MQZ_DELETE_AUTHORITY – Delete authority
7.4.7.1. QMgrName (MQCHAR48) – input
7.4.7.2. ObjectName (MQCHAR48) – input
7.4.7.3. ObjectType (MQLONG) – input
7.4.7.4. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.7.5. Continuation (MQLONG) – output
7.4.7.6. CompCode (MQLONG) – output
7.4.7.7. Reason (MQLONG) – output
7.4.8. MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data
7.4.8.1. QMgrName (MQCHAR48) – input
7.4.8.2. StartEnumeration (MQLONG) – input
7.4.8.3. Filter (MQZAD) – input
7.4.8.4. AuthorityBufferLength (MQLONG) – input
7.4.8.5. AuthorityBuffer (MQZAD) – output
7.4.8.6. AuthorityDataLength (MQLONG) – output
7.4.8.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.8.8. Continuation (MQLONG) – output
7.4.8.9. CompCode (MQLONG) – output
7.4.8.10. Reason (MQLONG) – output
7.4.9. MQZ_FREE_USER – Free user
7.4.9.1. QMgrName (MQCHAR48) – input

7.4.9.2. FreeParms (MQZFP) – input
7.4.9.3. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.9.4. Continuation (MQLONG) – output
7.4.9.5. CompCode (MQLONG) – output
7.4.9.6. Reason (MQLONG) – output
7.4.10. MQZ_GET_AUTHORITY – Get authority
7.4.10.1. QMgrName (MQCHAR48) – input
7.4.10.2. EntityName (MQCHAR12) – input
7.4.10.3. EntityType (MQLONG) – input
7.4.10.4. ObjectName (MQCHAR48) – input
7.4.10.5. ObjectType (MQLONG) – input
7.4.10.6. Authority (MQLONG) – output
7.4.10.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.10.8. Continuation (MQLONG) – output
7.4.10.9. CompCode (MQLONG) – output
7.4.10.10. Reason (MQLONG) – output
7.4.11. MQZ_GET_AUTHORITY_2 – Get authority (extended)
7.4.11.1. QMgrName (MQCHAR48) – input
7.4.11.2. EntityData (MQZED) – input
7.4.11.3. EntityType (MQLONG) – input
7.4.11.4. ObjectName (MQCHAR48) – input
7.4.11.5. ObjectType (MQLONG) – input
7.4.11.6. Authority (MQLONG) – output
7.4.11.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.11.8. Continuation (MQLONG) – output
7.4.11.9. CompCode (MQLONG) – output
7.4.11.10. Reason (MQLONG) – output
7.4.12. MQZ_GET_EXPLICIT_AUTHORITY – Get explicit authority
7.4.12.1. QMgrName (MQCHAR48) – input
7.4.12.2. EntityName (MQCHAR12) – input
7.4.12.3. EntityType (MQLONG) – input
7.4.12.4. ObjectName (MQCHAR48) – input
7.4.12.5. ObjectType (MQLONG) – input
7.4.12.6. Authority (MQLONG) – output
7.4.12.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.12.8. Continuation (MQLONG) – output
7.4.12.9. CompCode (MQLONG) – output
7.4.12.10. Reason (MQLONG) – output
7.4.13. MQZ_GET_EXPLICIT_AUTHORITY_2 – Get explicit authority (extended)
7.4.13.1. QMgrName (MQCHAR48) – input
7.4.13.2. EntityData (MQZED) – input
7.4.13.3. EntityType (MQLONG) – input
7.4.13.4. ObjectName (MQCHAR48) – input
7.4.13.5. ObjectType (MQLONG) – input
7.4.13.6. Authority (MQLONG) – output
7.4.13.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.13.8. Continuation (MQLONG) – output
7.4.13.9. CompCode (MQLONG) – output
7.4.13.10. Reason (MQLONG) – output
7.4.14. MQZ_INIT_AUTHORITY – Initialize authorization service
7.4.14.1. Hconfig (MQHCONFIG) – input
7.4.14.2. Options (MQLONG) – input
7.4.14.3. QMgrName (MQCHAR48) – input
7.4.14.4. ComponentDataLength (MQLONG) – input
7.4.14.5. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.14.6. Version (MQLONG) – input/output
7.4.14.7. CompCode (MQLONG) – output
7.4.14.8. Reason (MQLONG) – output
7.4.15. MQZ_INQUIRE – Inquire authorization service
7.4.15.1. QMgrName (MQCHAR48) – input
7.4.15.2. SelectorCount (MQLONG) – input
7.4.15.3. Selectors (MQLONG×SelectorCount) – input
7.4.15.4. IntAttrCount (MQLONG) – input
7.4.15.5. IntAttrs (MQLONG×IntAttrCount) – output
7.4.15.6. CharAttrCount (MQLONG) – input
7.4.15.7. CharAttrs (MQLONG×CharAttrCount) – output
7.4.15.8. SelectorReturned (MQLONG×SelectorCount) – input
7.4.15.9. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.15.10. Continuation (MQLONG) – output
7.4.15.11. CompCode (MQLONG) – output
7.4.15.12. Reason (MQLONG) – output
7.4.16. MQZ_REFRESH_CACHE – Refresh all authorizations
7.4.16.1. QMgrName (MQCHAR48) - input
7.4.16.2. ComponentData (MQBYTEžComponentDataLength) - input/output
7.4.16.3. Continuation (MQLONG) - output
7.4.16.4. CompCode (MQLONG) - output
7.4.16.5. Reason (MQLONG) - output
7.4.17. MQZ_SET_AUTHORITY – Set authority
7.4.17.1. QMgrName (MQCHAR48) – input
7.4.17.2. EntityName (MQCHAR12) – input
7.4.17.3. EntityType (MQLONG) – input
7.4.17.4. ObjectName (MQCHAR48) – input
7.4.17.5. ObjectType (MQLONG) – input
7.4.17.6. Authority (MQLONG) – input
7.4.17.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.17.8. Continuation (MQLONG) – output
7.4.17.9. CompCode (MQLONG) – output
7.4.17.10. Reason (MQLONG) – output
7.4.18. MQZ_SET_AUTHORITY_2 – Set authority (extended)
7.4.18.1. QMgrName (MQCHAR48) – input
7.4.18.2. EntityData (MQZED) – input
7.4.18.3. EntityType (MQLONG) – input
7.4.18.4. ObjectName (MQCHAR48) – input
7.4.18.5. ObjectType (MQLONG) – input
7.4.18.6. Authority (MQLONG) – input
7.4.18.7. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.18.8. Continuation (MQLONG) – output
7.4.18.9. CompCode (MQLONG) – output

7.4.18.10. Reason (MQLONG) – output
7.4.19. MQZ_TERM_AUTHORITY – Terminate authorization service
7.4.19.1. Hconfig (MQHCONFIG) – input
7.4.19.2. Options (MQLONG) – input
7.4.19.3. QMgrName (MQCHAR48) – input
7.4.19.4. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.19.5. CompCode (MQLONG) – output
7.4.19.6. Reason (MQLONG) – output
7.4.20. MQZAC – Application context
7.4.20.1. StrucId (MQCHAR4)
7.4.20.2. Version (MQLONG)
7.4.20.3. ProcessId (MQPID)
7.4.20.4. ThreadId (MQTID)
7.4.20.5. ApplName (MQCHAR28)
7.4.20.6. UserID (MQCHAR12)
7.4.20.7. EffectiveUserID (MQCHAR12)
7.4.20.8. Environment (MQLONG)
7.4.20.9. CallerType (MQLONG)
7.4.20.10. AuthenticationType (MQLONG)
7.4.20.11. BindType (MQLONG)
7.4.21. MQZAD – Authority data
7.4.21.1. StrucId (MQCHAR4)
7.4.21.2. Version (MQLONG)
7.4.21.3. ProfileName (MQCHAR48)
7.4.21.4. ObjectType (MQLONG)
7.4.21.5. Authority (MQLONG)
7.4.21.6. EntityDataPtr (PMQZED)
7.4.21.7. EntityType (MQLONG)
7.4.21.8. Options (MQAUTHOPT)
7.4.22. MQZED – Entity descriptor
7.4.22.1. StrucId (MQCHAR4)
7.4.22.2. Version (MQLONG)
7.4.22.3. EntityNamePtr (PMQCHAR)
7.4.22.4. EntityDomainPtr (PMQCHAR)
7.4.22.5. SecurityId (MQBYTE40)
7.4.22.6. CorrelationPtr (MQPTR)
7.4.23. MQZIC – Identity context
7.4.23.1. StrucId (MQCHAR4)
7.4.23.2. Version (MQLONG)
7.4.23.3. UserIdentifier (MQCHAR12)
7.4.23.4. AccountingToken (MQBYTE32)
7.4.23.5. ApplIdentityData (MQCHAR32)
7.4.24. MQZFP – Free parameters
7.4.24.1. StrucId (MQCHAR4)
7.4.24.2. Version (MQLONG)
7.4.24.3. Reserved (MQBYTE8)
7.4.24.4. CorrelationPtr (MQPTR)
7.4.25. MQZ_DELETE_NAME – Delete name
7.4.25.1. QMgrName (MQCHAR48) – input
7.4.25.2. QName (MQCHAR48) – input
7.4.25.3. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.25.4. Continuation (MQLONG) – output
7.4.25.5. CompCode (MQLONG) – output
7.4.25.6. Reason (MQLONG) – output
7.4.26. MQZ_INIT_NAME – Initialize name service
7.4.26.1. Hconfig (MQHCONFIG) – input
7.4.26.2. Options (MQLONG) – input
7.4.26.3. QMgrName (MQCHAR48) – input
7.4.26.4. ComponentDataLength (MQLONG) – input
7.4.26.5. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.26.6. Version (MQLONG) – input/output
7.4.26.7. CompCode (MQLONG) – output
7.4.26.8. Reason (MQLONG) – output
7.4.27. MQZ_INSERT_NAME – Insert name
7.4.27.1. QMgrName (MQCHAR48) – input
7.4.27.2. QName (MQCHAR48) – input
7.4.27.3. ResolvedQMgrName (MQCHAR48) – input
7.4.27.4. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.27.5. Continuation (MQLONG) – output
7.4.27.6. CompCode (MQLONG) – output
7.4.27.7. Reason (MQLONG) – output
7.4.28. MQZ_LOOKUP_NAME – Lookup name
7.4.28.1. QMgrName (MQCHAR48) – input
7.4.28.2. QName (MQCHAR48) – input
7.4.28.3. ResolvedQMgrName (MQCHAR48) – output
7.4.28.4. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.28.5. Continuation (MQLONG) – output
7.4.28.6. CompCode (MQLONG) – output
7.4.28.7. Reason (MQLONG) – output
7.4.29. MQZ_TERM_NAME – Terminate name service
7.4.29.1. Hconfig (MQHCONFIG) – input
7.4.29.2. Options (MQLONG) – input
7.4.29.3. QMgrName (MQCHAR48) – input
7.4.29.4. ComponentData (MQBYTE×ComponentDataLength) – input/output
7.4.29.5. CompCode (MQLONG) – output
7.4.29.6. Reason (MQLONG) – output
7.5. API exits
7.5.1. Why use API exits
7.5.2. How you use API exits
7.5.2.1. How to configure WebSphere MQ for API exits
7.5.2.2. How to write an API exit
7.5.3. What happens when an API exit runs?
7.5.4. Configuring API exits
7.5.4.1. Configuring API exits on UNIX systems
7.5.4.1.1. Attributes for all stanzas
7.5.4.1.2. Sample stanzas
7.5.4.1.3. Changing the configuration information
7.5.4.2. Configuring API exits on Windows systems

7.6. API exit reference information
7.6.1. General usage notes
7.6.2. MQACH – API exit chain header
7.6.2.1. Fields
7.6.2.1.1. StrucId (MQCHAR4)
7.6.2.1.2. Version (MQLONG)
7.6.2.1.3. StrucLength (MQLONG)
7.6.2.1.4. ChainAreaLength (MQLONG)
7.6.2.1.5. ExitInfoName (MQCHAR48)
7.6.2.1.6. NextChainAreaPtr (PMQACH)
7.6.2.2. C declaration
7.6.3. MQAXC – API exit context
7.6.3.1. Fields
7.6.3.1.1. StrucId (MQCHAR4)
7.6.3.1.2. Version (MQLONG)
7.6.3.1.3. Environment (MQLONG)
7.6.3.1.4. UserId (MQCHAR12)
7.6.3.1.5. SecurityId (MQBYTE40)
7.6.3.1.6. ConnectionName (MQCHAR264)
7.6.3.1.7. LongMCAUserIdLength (MQLONG)
7.6.3.1.8. LongRemoteUserIdLength (MQLONG)
7.6.3.1.9. LongMCAUserIdPtr (MQPTR)
7.6.3.1.10. LongRemoteUserIdPtr (MQPTR)
7.6.3.1.11. ApplName (MQCHAR28)
7.6.3.1.12. ApplType (MQLONG)
7.6.3.1.13. ProcessId (MQPID)
7.6.3.1.14. ThreadId (MQTID)
7.6.3.2. C declaration
7.6.4. MQAXP – API exit parameter
7.6.4.1. Fields
7.6.4.1.1. StrucId (MQCHAR4)
7.6.4.1.2. Version (MQLONG)
7.6.4.1.3. ExitId (MQLONG)
7.6.4.1.4. ExitReason (MQLONG)
7.6.4.1.5. ExitResponse (MQLONG)
7.6.4.1.6. ExitResponse2 (MQLONG)
7.6.4.1.7. Feedback (MQLONG)
7.6.4.1.8. APICallerType (MQLONG)
7.6.4.1.9. ExitUserArea (MQBYTE16)
7.6.4.1.10. ExitData (MQCHAR32)
7.6.4.1.11. ExitInfoName (MQCHAR48)
7.6.4.1.12. ExitPDArea (MQBYTE48)
7.6.4.1.13. QMgrName (MQCHAR48)
7.6.4.1.14. ExitChainAreaPtr (PMQACH)
7.6.4.1.15. Hconfig (MQHCONFIG)
7.6.4.1.16. Function (MQLONG)
7.6.4.1.17. ExitMsgHandle (MQHMSG)
7.6.4.2. C declaration
7.6.5. MQXEP – Register entry point
7.6.5.1. Syntax
7.6.5.2. Parameters
7.6.5.2.1. Hconfig (MQHCONFIG) – input
7.6.5.2.2. ExitReason (MQLONG) – input
7.6.5.2.3. Function (MQLONG) – input
7.6.5.2.4. EntryPoint (PMQFUNC) – input
7.6.5.2.5. ExitOpts (MQXEPO) – input
7.6.5.2.6. pCompCode (PMQLONG) – output
7.6.5.2.7. pReason (PMQLONG) – output
7.6.5.3. C invocation
7.6.6. MQXEPO Register entry point options
7.6.6.1. Overview
7.6.6.2. Fields
7.6.6.2.1. ExitProperties (MQCHARV)
7.6.6.2.2. Options (MQLONG)
7.6.6.2.3. StrucId (MQCHAR4)
7.6.6.2.4. Version (MQLONG)
7.6.6.3. Language declarations
7.6.6.3.1. C declaration
7.6.7. MQNXP structure - Set by the caller of the exit
7.6.8. MQ_BACK_EXIT – Back out changes
7.6.8.1. Syntax
7.6.8.2. Parameters
7.6.8.2.1. pExitParms (PMQAXP) – input/output
7.6.8.2.2. pExitContext (PMQAXC) – input/output
7.6.8.2.3. pHconn (PMQHCONN) – input/output
7.6.8.2.4. pCompCode (PMQLONG) – input/output
7.6.8.2.5. pReason (PMQLONG) – input/output
7.6.8.3. C invocation
7.6.9. MQ_BEGIN_EXIT – Begin unit of work
7.6.9.1. Syntax
7.6.9.2. Parameters
7.6.9.2.1. pExitParms (PMQAXP) – input/output
7.6.9.2.2. pExitContext (PMQAXC) – input/output
7.6.9.2.3. pHconn (PMQHCONN) – input/output
7.6.9.2.4. ppBeginOptions (PPMQBO) – input/output
7.6.9.2.5. pCompCode (PMQLONG) – input/output
7.6.9.2.6. pReason (PMQLONG) – input/output
7.6.9.3. C invocation
7.6.10. MQ_CALLBACK_EXIT - Callback
7.6.10.1. Syntax
7.6.10.2. Parameters
7.6.10.2.1. pExitParms (PMQAXP) – input/output
7.6.10.2.2. pExitContext (PMQAXC) – input/output
7.6.10.2.3. pHconn (PMQHCONN) – input/output
7.6.10.2.4. ppHobj (PPMQHOBJ) – input/output
7.6.10.2.5. ppMsgDesc (PPMQMD) – input/output
7.6.10.2.6. ppGetMsgOpts (PPMQGMO) – input/output
7.6.10.2.7. ppBuffer (PPMQVOID) – input/output

7.6.10.2.8. ppContext (PPMQCBC) – input/output
7.6.10.3. Usage notes
7.6.10.4. C invocation
7.6.11. MQ_CLOSE_EXIT – Close object
7.6.11.1. Syntax
7.6.11.2. Parameters
7.6.11.2.1. pExitParms (PMQAXP) – input/output
7.6.11.2.2. pExitContext (PMQAXC) – input/output
7.6.11.2.3. pHconn (PMQHCONN) – input/output
7.6.11.2.4. ppHobj (PPMQHOBJ) – input/output
7.6.11.2.5. pOptions (PMQLONG) – input/output
7.6.11.2.6. pCompCode (PMQLONG) – input/output
7.6.11.2.7. pReason (PMQLONG) – input/output
7.6.11.3. C invocation
7.6.12. MQ_CMIT_EXIT – Commit changes
7.6.12.1. Syntax
7.6.12.2. Parameters
7.6.12.2.1. pExitParms (PMQAXP) – input/output
7.6.12.2.2. pExitContext (PMQAXC) – input/output
7.6.12.2.3. pHconn (PMQHCONN) – input/output
7.6.12.2.4. pCompCode (PMQLONG) – input/output
7.6.12.2.5. pReason (PMQLONG) – input/output
7.6.12.3. C invocation
7.6.13. MQ_CONNX_EXIT – Connect queue manager (extended)
7.6.13.1. Syntax
7.6.13.2. Parameters
7.6.13.2.1. pExitParms (PMQAXP) – input/output
7.6.13.2.2. pExitContext (PMQAXC) – input/output
7.6.13.2.3. pQMgrName (PMQCHAR48) – input/output
7.6.13.2.4. ppConnectOpts (PPMQCNO) – input/output
7.6.13.2.5. ppHconn (PPMQHCONN) – input/output
7.6.13.2.6. pCompCode (PMQLONG) – input/output
7.6.13.2.7. pReason (PMQLONG) – input/output
7.6.13.3. Usage notes
7.6.13.4. C invocation
7.6.14. MQ_DISC_EXIT – Disconnect queue manager
7.6.14.1. Syntax
7.6.14.2. Parameters
7.6.14.2.1. pExitParms (PMQAXP) – input/output
7.6.14.2.2. pExitContext (PMQAXC) – input/output
7.6.14.2.3. ppHconn (PPMQHCONN) – input/output
7.6.14.2.4. pCompCode (PMQLONG) – input/output
7.6.14.2.5. pReason (PMQLONG) – input/output
7.6.14.3. C invocation
7.6.15. MQ_GET_EXIT – Get message
7.6.15.1. Syntax
7.6.15.2. Parameters
7.6.15.2.1. pExitParms (PMQAXP) – input/output
7.6.15.2.2. pExitContext (PMQAXC) – input/output
7.6.15.2.3. pHconn (PMQHCONN) – input/output
7.6.15.2.4. pHobj (PMQHOBJ) – input/output
7.6.15.2.5. ppMsgDesc (PPMQMD) – input/output
7.6.15.2.6. ppGetMsgOpts (PPMQGMO) – input/output
7.6.15.2.7. pBufferLength (PMQLONG) – input/output
7.6.15.2.8. ppBuffer (PPMQVOID) – input/output
7.6.15.2.9. ppDataLength (PPMQLONG) – input/output
7.6.15.2.10. pCompCode (PMQLONG) – input/output
7.6.15.2.11. pReason (PMQLONG) – input/output
7.6.15.3. Usage notes
7.6.15.4. C invocation
7.6.16. MQ_INIT_EXIT – Initialize exit environment
7.6.16.1. Syntax
7.6.16.2. Parameters
7.6.16.2.1. pExitParms (PMQAXP) – input/output
7.6.16.2.2. pExitContext (PMQAXC) – input/output
7.6.16.2.3. pCompCode (PMQLONG) – input/output
7.6.16.2.4. pReason (PMQLONG) – input/output
7.6.16.3. Usage notes
7.6.16.4. C invocation
7.6.17. MQ_INQ_EXIT – Inquire object attributes
7.6.17.1. Syntax
7.6.17.2. Parameters
7.6.17.2.1. pExitParms (PMQAXP) – input/output
7.6.17.2.2. pExitContext (PMQAXC) – input/output
7.6.17.2.3. pHconn (PMQHCONN) – input/output
7.6.17.2.4. pHobj (PMQHOBJ) – input/output
7.6.17.2.5. pSelectorCount (PMQLONG) – input/output
7.6.17.2.6. ppSelectors (PPMQLONG) – input/output
7.6.17.2.7. pIntAttrCount (PMQLONG) – input/output
7.6.17.2.8. ppIntAttrs (PPMQLONG) – input/output
7.6.17.2.9. pCharAttrLength (PMQLONG) – input/output
7.6.17.2.10. ppCharAttrs (PPMQCHAR) – input/output
7.6.17.2.11. pCompCode (PMQLONG) – input/output
7.6.17.2.12. pReason (PMQLONG) – input/output
7.6.17.3. C invocation
7.6.18. MQ_OPEN_EXIT – Open object
7.6.18.1. Syntax
7.6.18.2. Parameters
7.6.18.2.1. pExitParms (PMQAXP) – input/output
7.6.18.2.2. pExitContext (PMQAXC) – input/output
7.6.18.2.3. pHconn (PMQHCONN) – input/output
7.6.18.2.4. ppObjDesc (PPMQOD) – input/output
7.6.18.2.5. pOptions (PMQLONG) – input/output
7.6.18.2.6. ppHobj (PPMQHOBJ) – input/output
7.6.18.2.7. pCompCode (PMQLONG) – input/output
7.6.18.2.8. pReason (PMQLONG) – input/output
7.6.18.3. C invocation
7.6.19. MQ_PUT_EXIT – Put message

7.6.19.1. Syntax
7.6.19.2. Parameters
7.6.19.2.1. pExitParms (PMQAXP) – input/output
7.6.19.2.2. pExitContext (PMQAXC) – input/output
7.6.19.2.3. pHconn (PMQHCONN) – input/output
7.6.19.2.4. pHobj (PMQHOBJ) – input/output
7.6.19.2.5. ppMsgDesc (PPMQMD) – input/output
7.6.19.2.6. ppPutMsgOpts (PPMQPMO) – input/output
7.6.19.2.7. pBufferLength (PMQLONG) – input/output
7.6.19.2.8. ppBuffer (PPMQVOID) – input/output
7.6.19.2.9. pCompCode (PMQLONG) – input/output
7.6.19.2.10. pReason (PMQLONG) – input/output
7.6.19.3. Usage notes
7.6.19.4. C invocation
7.6.20. MQ_PUT1_EXIT – Put one message
7.6.20.1. Syntax
7.6.20.2. Parameters
7.6.20.2.1. pExitParms (PMQAXP) – input/output
7.6.20.2.2. pExitContext (PMQAXC) – input/output
7.6.20.2.3. pHconn (PMQHCONN) – input/output
7.6.20.2.4. ppObjDesc (PPMQOD) – input/output
7.6.20.2.5. ppMsgDesc (PPMQMD) – input/output
7.6.20.2.6. ppPutMsgOpts (PPMQPMO) – input/output
7.6.20.2.7. pBufferLength (PMQLONG) – input/output
7.6.20.2.8. ppBuffer (PPMQVOID) – input/output
7.6.20.2.9. pCompCode (PMQLONG) – input/output
7.6.20.2.10. pReason (PMQLONG) – input/output
7.6.20.3. C invocation
7.6.21. MQ_SET_EXIT – Set object attributes
7.6.21.1. Syntax
7.6.21.2. Parameters
7.6.21.2.1. pExitParms (PMQAXP) – input/output
7.6.21.2.2. pExitContext (PMQAXC) – input/output
7.6.21.2.3. pHconn (PMQHCONN) – input/output
7.6.21.2.4. pHobj (PMQHOBJ) – input/output
7.6.21.2.5. pSelectorCount (PMQLONG) – input/output
7.6.21.2.6. ppSelectors (PPMQLONG) – input/output
7.6.21.2.7. pIntAttrCount (PMQLONG) – input/output
7.6.21.2.8. ppIntAttrs (PPMQLONG) – input/output
7.6.21.2.9. pCharAttrLength (PMQLONG) – input/output
7.6.21.2.10. ppCharAttrs (PPMQCHAR) – input/output
7.6.21.2.11. pCompCode (PMQLONG) – input/output
7.6.21.2.12. pReason (PMQLONG) – input/output
7.6.21.3. C invocation
7.6.22. MQ_SUB_EXIT – Register subscription
7.6.22.1. Syntax
7.6.22.2. Parameters
7.6.22.2.1. pExitParms (PMQAXP) – input/output
7.6.22.2.2. pExitContext (PMQAXC) – input/output
7.6.22.2.3. pHConn (PMQHCONN) – input/output
7.6.22.2.4. ppSubDesc (PPMQSD) – input/output
7.6.22.2.5. ppHobj (PPMQHOBJ) – input/output
7.6.22.2.6. ppHsub (PPMQHOBJ) – input/output
7.6.22.2.7. pCompCode (PMQLONG) – input/output
7.6.22.2.8. pReason (PMQLONG) – input/output
7.6.22.3. C invocation
7.6.23. MQ_SUBRQ_EXIT – Subscription request
7.6.23.1. Syntax
7.6.23.2. Parameters
7.6.23.2.1. pExitParms (PMQAXP) – input/output
7.6.23.2.2. pExitContext (PMQAXC) – input/output
7.6.23.2.3. pHConn (PMQHCONN) – input/output
7.6.23.2.4. ppHsub (PPMQHOBJ) – input/output
7.6.23.2.5. pAction (PMQLONG) – input/output
7.6.23.2.6. ppSubRqOpts (PPMQSRO) – input/output
7.6.23.2.7. pCompCode (PMQLONG) – input/output
7.6.23.2.8. pReason (PMQLONG) – input/output
7.6.23.3. C invocation
7.6.24. MQ_TERM_EXIT – Terminate exit environment
7.6.24.1. Syntax
7.6.24.2. Parameters
7.6.24.2.1. pExitParms (PMQAXP) – input/output
7.6.24.2.2. pExitContext (PMQAXC) – input/output
7.6.24.2.3. pCompCode (PMQLONG) – input/output
7.6.24.2.4. pReason (PMQLONG) – input/output
7.6.24.3. Usage notes
7.6.24.4. C invocation
7.7. Referencing connection definitions using a pre-connect exit from a repository
8. File Transfer Application
8.1. Introduction
8.1.1. Advantages
8.1.2. Components
8.2. Installing and configuring
8.2.1. Installing the File Transfer Application on a WebSphere MQ server
8.2.1.1. During the initial installation
8.2.1.2. Modifying the installation
8.2.2. Installing the File Transfer Application on a WebSphere MQ client
8.2.2.1. During the initial installation
8.2.2.2. Modifying the installation
8.2.3. Setup tasks
8.2.3.1. Sending files between remote queue managers
8.2.3.2. Sending files between a queue manager and a remote WebSphere MQ client
8.2.4. Configuring the GUI
8.2.5. File Transfer Application channel security
8.3. Using the File Transfer Application
8.3.1. Sending a file
8.3.2. Receiving a file
8.3.3. Listing all sent and received files

8.3.4. File status
8.3.5. Using the command line

System Administration Guide

Introduction

Administration using WebSphere MQ commands

Administration using the WebSphere MQ Explorer
The WebSphere® MQ Explorer allows you to perform local or remote administration of your network from a computer running Windows, or Linux (x86 platform)

Using the WebSphere MQ Taskbar application (Windows only)
The WebSphere MQ Taskbar application displays an icon in the Windows system tray on the server. The icon provides you with the current status of WebSphere MQ and a context menu from which you can perform some simple actions.

WebSphere MQ control commands

Configuring and managing WebSphere MQ
Configure WebSphere MQ to meet your organization needs, perform operational tasks, and solve operational problems.

WebSphere MQ installable services and the API exit

File Transfer Application

Notices

This build: January 26, 2011 11:54:25

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10120_

1. Introduction

Introduction to WebSphere MQ

An introduction to WebSphere MQ administration

Parent topic: System Administration Guide

This build: January 26, 2011 11:54:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10260_

1.1. Introduction to WebSphere MQ

For WebSphere® MQ on z/OS system administration information, see WebSphere MQ z/OS System Administration Guide ; for WebSphere MQ on i5/OS system administration information, see WebSphere MQ on i5/OS System Administration Guide.

This section introduces WebSphere MQ Version 7 from an administrator's perspective, and describes the basic concepts of WebSphere MQ and messaging on UNIX and Windows systems. It contains these sections:

WebSphere MQ and message queuing

Messages and queues

Objects

Clients and servers

Extending queue manager facilities

Security

Transactional support

Parent topic: Introduction

Related information
WebSphere MQ on z/OS System Administration Guide
WebSphere MQ on i5/OS System Administration Guide

This build: January 26, 2011 11:54:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10270_

1.1.1. WebSphere MQ and message queuing

WebSphere® MQ allows application programs to use *message queuing* to participate in message-driven processing. Application programs can communicate across different platforms by using the appropriate message queuing software products. For example, HP-UX and z/OS® applications can communicate through WebSphere MQ for HP-UX and WebSphere MQ for z/OS respectively. The applications are shielded from the mechanics of the underlying communications.

WebSphere MQ implements a common application programming interface known as the *message queue interface* (or MQI) wherever the applications run. This makes it easier for you to port application programs from one platform to another.

The MQI is described in detail in the WebSphere MQ Application Programming Reference.

Time-independent applications

Message-driven processing

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10280_

1.1.1.1. Time-independent applications

With message queuing, the exchange of messages between the sending and receiving programs is independent of time. This means that the sending and receiving application programs are decoupled; the sender can continue processing without having to wait for the receiver to acknowledge receipt of the message. The target application does not even have to be running when the message is sent. It can retrieve the message after it is has been started.

Parent topic: WebSphere MQ and message queuing

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10290_

1.1.1.2. Message-driven processing

When messages arrive on a queue, they can automatically start an application using *triggering*. If necessary, the applications can be stopped when the message (or messages) have been processed.

Parent topic: WebSphere MQ and message queuing

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10300_

1.1.2. Messages and queues

Messages and queues are the basic components of a message queuing system.

What is a message?

What is a queue?

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10310_

1.1.2.1. What is a message?

A *message* is a string of bytes that is meaningful to the applications that use it. Messages are used to transfer information from one application program to another (or between different parts of the same application). The applications can be running on the same platform, or on different platforms.

WebSphere® MQ messages have two parts:

- *The application data.* The content and structure of the application data is defined by the application programs that use it.
- *A message descriptor.* The message descriptor identifies the message and contains additional control information, such as the type of message and the priority assigned to the message by the sending application.
  The format of the message descriptor is defined by WebSphere MQ. For a complete description of the message descriptor, see the WebSphere MQ Application Programming Reference manual.

Message lengths

How do applications send and receive messages?

Parent topic: Messages and queues

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10320_

1.1.2.1.1. Message lengths

The default maximum message length is 4 MB, although you can increase this to a maximum length of 100 MB (where 1 MB equals 1 048 576 bytes). In practice, the message length might be limited by:

- The maximum message length defined for the receiving queue
- The maximum message length defined for the queue manager
- The maximum message length defined by the queue
- The maximum message length defined by either the sending or receiving application
- The amount of storage available for the message

It might take several messages to send all the information that an application requires.

Parent topic: What is a message?

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10330_

1.1.2.1.2. How do applications send and receive messages?

Application programs send and receive messages using **MQI calls**.

For example, to put a message onto a queue, an application:

1. Opens the required queue by issuing an MQI **MQOPEN** call
2. Issues an MQI **MQPUT** call to put the message onto the queue

Another application can retrieve the message from the same queue by issuing an MQI **MQGET** call

For more information about MQI calls, see the WebSphere MQ Application Programming Reference.

Parent topic: What is a message?

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10340_

1.1.2.2. What is a queue?

A *queue* is a data structure used to store messages.

Each queue is owned by a *queue manager*. The queue manager is responsible for maintaining the queues it owns, and for storing all the messages it receives onto the appropriate queues. The messages might be put on the queue by application programs, or by a queue manager as part of its normal operation.

WebSphere® MQ Version 7.0 supports queues over 2 GB in size; Enabling large queues discusses this in more detail. For information about planning the amount of storage you need for queues, see the *Quick Beginnings* guide for your platform, or visit the WebSphere MQ Web site for platform-specific performance reports:

```
http://www.ibm.com/software/integration/ts/mqseries/
```

Predefined queues and dynamic queues

Retrieving messages from queues

Parent topic: Messages and queues

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10350_

1.1.2.2.1. Predefined queues and dynamic queues

Queues can be characterized by the way they are created:

- **Predefined queues** are created by an administrator using the appropriate MQSC or PCF commands. Predefined queues are permanent; they exist independently of the applications that use them and survive WebSphere® MQ restarts.
- **Dynamic queues** are created when an application issues an **MQOPEN** request specifying the name of a *model queue*. The queue created is based on a *template queue definition*, which is called a model queue. You can create a model queue using the MQSC command DEFINE QMODEL. The attributes of a model queue (for example, the maximum number of messages that can be stored on it) are inherited by any dynamic queue that is created from it.
  Model queues have an attribute that specifies whether the dynamic queue is to be permanent or temporary. Permanent queues survive application and

queue manager restarts; temporary queues are lost on restart.

Parent topic: What is a queue?

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10360_

1.1.2.2.2. Retrieving messages from queues

Suitably authorized applications can retrieve messages from a queue according to the following retrieval algorithms:

- First-in-first-out (FIFO).
- Message priority, as defined in the message descriptor. Messages that have the same priority are retrieved on a FIFO basis.
- A program request for a specific message.

The **MQGET** request from the application determines the method used.

Parent topic: What is a queue?

This build: January 26, 2011 11:54:26

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10370_

1.1.3. Objects

Many of the tasks described in this book involve manipulating WebSphere® MQ **objects**. The object types are queue managers, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects.

The manipulation or *administration* of objects includes:

- Starting and stopping queue managers.
- Creating objects, particularly queues, for applications.
- Working with channels to create communication paths to queue managers on other (remote) systems. This is described in detail in WebSphere MQ Intercommunication.
- Creating *clusters* of queue managers to simplify the overall administration process, and to balance workload.

This book contains detailed information about administration in the following chapters:

- An introduction to WebSphere MQ administration
- Managing queue managers
- Administering local WebSphere MQ objects
- Automating administration tasks
- Administering remote WebSphere MQ objects
- Administration using the WebSphere MQ Explorer

Object names

Managing objects

Object attributes

WebSphere MQ queues
There are four types of queue object available in WebSphere MQ; the local queue object, the remote queue object, the alias queue object, and the model queue object.

WebSphere MQ queue managers

Process definitions

Clusters

Namelists

Authentication information objects

Channels

Client connection channels

Listeners

Services

Administrative topic objects
An *administrative topic object* is a WebSphere MQ object that allows you to assign specific, non-default attributes to topics.

System default objects

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10380_

1.1.3.1. Object names

The naming convention adopted for WebSphere® MQ objects depends on the object.

Each instance of a queue manager is known by its name. This name must be unique within the network of interconnected queue managers, so that one queue manager can unambiguously identify the target queue manager to which any given message is sent.

For the other types of object, each object has a name associated with it and can be referred to by that name. These names must be unique within one queue manager and object type. For example, you can have a queue and a process with the same name, but you cannot have two queues with the same name.

In WebSphere MQ, names can have a maximum of 48 characters, with the exception of *channels* which have a maximum of 20 characters. For more information about names, see Names of WebSphere MQ objects.

Parent topic: Objects

This build: January 26, 2011 11:54:27

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10390_

1.1.3.2. Managing objects

You can create, alter, display, and delete objects using:

- Control commands, which are typed in from a keyboard
- MQSC commands, which can be typed in from a keyboard or read from a file
- Programmable Command Format (PCF) messages, which can be used in an automation program
- WebSphere® MQ Administration Interface (MQAI) calls in a program
- The WebSphere MQ Explorer
- WebSphere MQ for Windows only:
  - MQAI Component Object Model (COM) calls in a program
  - Active Directory Service interface (ADSI) calls in a program
  - The Windows Default Configuration Application

For more information about these methods, see An introduction to WebSphere MQ administration.

Parent topic: Objects

This build: January 26, 2011 11:54:27

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10400_

1.1.3.3. Object attributes

The properties of an object are defined by its attributes. Some you can specify, others you can only view. For example, the maximum message length that a queue can accommodate is defined by its `MaxMsgLength` attribute; you can specify this attribute when you create a queue. The `DefinitionType` attribute specifies how the queue was created; you can only display this attribute.

In WebSphere® MQ, there are two ways of referring to an attribute:

- Using its PCF name, for example, `MaxMsgLength`.
- Using its MQSC command name, for example, MAXMSGL.

This book mainly describes how to specify attributes using MQSC commands, and so it refers to most attributes using their MQSC command names, rather than their PCF names.

Parent topic: Objects

This build: January 26, 2011 11:54:27

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10410_

1.1.3.4. WebSphere MQ queues

There are four types of queue object available in WebSphere® MQ; the local queue object, the remote queue object, the alias queue object, and the model queue object.

These four types of queue object are described in the following section.

Local queue object

A local queue object identifies a local queue belonging to the queue manager to which the application is connected. All queues are local queues in the sense that each queue belongs to a queue manager and, for that queue manager, the queue is a local queue.

Remote queue object

A remote queue object identifies a queue belonging to another queue manager. This queue must be defined as a local queue to that queue manager. The information you specify when you define a remote queue object allows the local queue manager to find the remote queue manager, so that any messages destined for the remote queue go to the correct queue manager.

Before applications can send messages to a queue on another queue manager, you must have defined a transmission queue and channels between the queue managers, **unless** you have grouped one or more queue managers together into a *cluster*. For more information about clusters, see Remote administration using clusters.

Alias queue object

An alias queue allows applications to access a queue by referring to it indirectly in MQI calls. When an alias queue name is used in an MQI call, the name is resolved to the name of either a local or a remote queue at run time. This allows you to change the queues that applications use without changing the application in any way; you merely change the alias queue definition to reflect the name of the new queue to which the alias resolves.

An alias queue is not a queue, but an object that you can use to access another queue.

Model queue object

A model queue defines a set of queue attributes that are used as a template for creating a dynamic queue. Dynamic queues are created by the queue manager when an application issues an **MQOPEN** request specifying a queue name that is the name of a model queue. The dynamic queue that is created in this way is a local queue whose attributes are taken from the model queue definition. The dynamic queue name can be specified by the application, or the queue manager can generate the name and return it to the application.

Dynamic queues defined in this way can be temporary queues, which do not survive product restarts, or permanent queues, which do.

Defining queues

Queues used by WebSphere MQ

Parent topic: Objects

This build: January 26, 2011 11:54:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10420_

1.1.3.4.1. Defining queues

Queues are defined to WebSphere® MQ using:

- The MQSC command DEFINE
- The PCF Create Queue command

The commands specify the type of queue and its attributes. For example, a local queue object has attributes that specify what happens when applications reference that queue in MQI calls. Examples of attributes are:

- Whether applications can retrieve messages from the queue (GET enabled)
- Whether applications can put messages on the queue (PUT enabled)
- Whether access to the queue is exclusive to one application or shared between applications
- The maximum number of messages that can be stored on the queue at the same time (maximum queue depth)
- The maximum length of messages that can be put on the queue

For further details about defining queue objects, see the WebSphere MQ Script (MQSC) Command Reference or WebSphere MQ Programmable Command Formats and Administration Interface.

Parent topic: WebSphere MQ queues

This build: January 26, 2011 11:54:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10430_

1.1.3.4.2. Queues used by WebSphere MQ

WebSphere® MQ uses some local queues for specific purposes related to its operation. You **must** define these queues before WebSphere MQ can use them.

Initiation queues

Initiation queues are queues that are used in triggering. A queue manager puts a trigger message on an initiation queue when a trigger event occurs. A trigger event is a logical combination of conditions that is detected by a queue manager. For example, a trigger event might be generated when the number of messages on a queue reaches a predefined depth. This event causes the queue manager to put a trigger message on a specified initiation queue. This trigger message is retrieved by a *trigger monitor*, a special application that monitors an initiation queue. The trigger monitor then starts the application program that was specified in the trigger message.

If a queue manager is to use triggering, at least one initiation queue must be defined for that queue manager. See Managing objects for triggering and runmqtrm. For more information about triggering, see the WebSphere MQ Application Programming Guide.

Transmission queues

Transmission queues are queues that temporarily store messages that are destined for a remote queue manager. You must define at least one transmission queue for each remote queue manager to which the local queue manager is to send messages directly. These queues are also used in remote administration; see Remote administration from a local queue manager. For information about the use of transmission queues in distributed queuing, see WebSphere MQ Intercommunication.

Each queue manager can have a default transmission queue. When a queue manager that is not part of a cluster puts a message onto a remote queue, the default action, if there is no transmission queue with the same name as the destination queue manager, is to use the default transmission queue.

Cluster transmission queues

Each queue manager within a cluster has a cluster transmission queue called SYSTEM.CLUSTER.TRANSMIT.QUEUE. A definition of this queue is created by default when you define a queue manager.

A queue manager that is part of the cluster can send messages on the cluster transmission queue to any other queue manager that is in the same cluster.

During name resolution, the cluster transmission queue takes precedence over the default transmission queue.

When a queue manager is part of a cluster, the default action is to use the SYSTEM.CLUSTER.TRANSMIT.QUEUE, except when the destination queue manager is not part of the cluster.

Dead-letter queues

A dead-letter (undelivered-message) queue is a queue that stores messages that cannot be routed to their correct destinations. This occurs when, for example, the destination queue is full. The supplied dead-letter queue is called SYSTEM.DEAD.LETTER.QUEUE.

For distributed queuing, define a dead-letter queue on each queue manager involved.

Command queues

The command queue, SYSTEM.ADMIN.COMMAND.QUEUE, is a local queue to which suitably authorized applications can send MQSC commands for processing. These commands are then retrieved by a WebSphere MQ component called the command server. The command server validates the commands, passes the valid ones on for processing by the queue manager, and returns any responses to the appropriate reply-to queue.

A command queue is created automatically for each queue manager when that queue manager is created.

Reply-to queues

When an application sends a request message, the application that receives the message can send back a reply message to the sending application. This message is put on a queue, called a reply-to queue, which is normally a local queue to the sending application. The name of the reply-to queue is specified by the sending application as part of the message descriptor.

Event queues

Instrumentation events can be used to monitor queue managers independently of MQI applications.

When an instrumentation event occurs, the queue manager puts an event message on an event queue. This message can then be read by a monitoring application, which might inform an administrator or initiate some remedial action if the event indicates a problem.

Note: Trigger events are quite different from instrumentation events in that trigger events are not caused by the same conditions, and do not generate event messages.

For more information about instrumentation events, see Monitoring WebSphere MQ.

Parent topic: WebSphere MQ queues

This build: January 26, 2011 11:54:27

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10440_

1.1.3.5. WebSphere MQ queue managers

A *queue manager* provides queuing services to applications, and manages the queues that belong to it. It ensures that:

- Object attributes are changed according to the commands received.
- Special events such as trigger events or instrumentation events are generated when the appropriate conditions are met.
- Messages are put on the correct queue, as requested by the application making the **MQPUT** call. The application is informed if this cannot be done, and an appropriate reason code is given.

Each queue belongs to a single queue manager and is said to be a *local queue* to that queue manager. The queue manager to which an application is connected is said to be the *local queue manager* for that application. For the application, the queues that belong to its local queue manager are local queues.

A *remote queue* is a queue that belongs to another queue manager. A *remote queue manager* is any queue manager other than the local queue manager. A remote queue manager can exist on a remote machine across the network, or might exist on the same machine as the local queue manager. WebSphere® MQ supports multiple queue managers on the same machine.

A queue manager object can be used in some MQI calls. For example, you can inquire about the attributes of the queue manager object using the MQI call **MQINQ**.

Parent topic: Objects

This build: January 26, 2011 11:54:27

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10450_

1.1.3.6. Process definitions

A *process definition* object defines an application that starts in response to a trigger event on a WebSphere® MQ queue manager. See the "Initiation queues" entry under Queues used by WebSphere MQ for more information.

The process definition attributes include the application ID, the application type, and data specific to the application.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10460_

1.1.3.7. Clusters

In a traditional WebSphere® MQ network using distributed queuing, every queue manager is independent. If one queue manager needs to send messages to another queue manager, it must define a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

A *cluster* is a group of queue managers set up in such a way that the queue managers can communicate directly with one another over a single network,

without the need for transmission queue, channel, and remote queue definitions.

For information about clusters, see Administering remote WebSphere MQ objects, and WebSphere MQ Queue Manager Clusters.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10470_

1.1.3.8. Namelists

A namelist is a WebSphere® MQ object that contains a list of other WebSphere MQ objects. Typically, namelists are used by applications such as trigger monitors, where they are used to identify a group of queues. The advantage of using a namelist is that it is maintained independently of applications; it can be updated without stopping any of the applications that use it. Also, if one application fails, the namelist is not affected and other applications can continue using it.

Namelists are also used with queue manager clusters to maintain a list of clusters referred to by more than one WebSphere MQ object.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10480_

1.1.3.9. Authentication information objects

The queue manager authentication information object forms part of WebSphere® MQ support for Secure Sockets Layer (SSL) security. It provides the definitions needed to check for revoked certificates. Certification Authorities revoke certificates that can no longer be trusted.

This book describes using the **setmqaut**, **dspmqaut**, **dmpmqaut**, **rcrmqobj**, **rcdmqimg**, and **dspmqfls** commands with the authentication information object. For an overview of SSL and the use of the authentication information objects, see WebSphere MQ Security.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10490_

1.1.3.10. Channels

Channels are objects that provide a communication path from one queue manager to another. Channels are used in distributed queuing to move messages from one queue manager to another. They shield applications from the underlying communications protocols. The queue managers might exist on the same, or different, platforms. For queue managers to communicate with one another, you must define one channel object at the queue manager that is to send messages, and another, complementary one, at the queue manager that is to receive them.

For information on channels and how to use them, see WebSphere MQ Intercommunication.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10500_

1.1.3.11. Client connection channels

Client connection channels are objects that provide a communication path from a WebSphere® MQ client to a queue manager. Client connection channels are used in distributed queuing to move messages between a queue manager and a client. They shield applications from the underlying communications protocols. The client might exist on the same, or different, platform to the queue manager.

For information on client connection channels and how to use them, see WebSphere MQ Intercommunication.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10510_

1.1.3.12. Listeners

Listeners are processes that accept network requests from other queue managers, or client applications, and start associated channels. Listener processes can be started using the **runmqlsr** control command.

Listener objects are WebSphere® MQ objects that allow you to manage the starting and stopping of listener processes from within the scope of a queue manager. By defining attributes of a listener object you do the following:

- Configure the listener process.
- Specify whether the listener process automatically starts and stops when the queue manager starts and stops.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10520_

1.1.3.13. Services

Service objects are a way of defining programs to be executed when a queue manager starts or stops. The programs can be split into the following types:

Servers

A *server* is a service object that has the parameter SERVTYPE specified as SERVER. A server service object is the definition of a program that will be executed when a specified queue manager is started. Only one instance of a server process can be executed concurrently. While running, the status of a server process can be monitored using the MQSC command, DISPLAY SVSTATUS. Typically server service objects are definitions of programs such as dead letter handlers or trigger monitors, however the programs that can be run are not limited to those supplied with WebSphere® MQ. Additionally, a server service object can be defined to include a command that will be run when the specified queue manager is shutdown to end the program.

Commands

A *command* is a service object that has the parameter SERVTYPE specified as COMMAND. A command service object is the definition of a program that will be executed when a specified queue manager is started or stopped. Multiple instances of a command process can be executed concurrently. Command service objects differ from server service objects in that once the program is executed the queue manager will not monitor the program. Typically command service objects are definitions of programs that are short lived and will perform a specific task such as starting one, or more, other tasks.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10530_

1.1.3.14. Administrative topic objects

An *administrative topic object* is a WebSphere® MQ object that allows you to assign specific, non-default attributes to topics.

A *topic* is defined by an application publishing or subscribing to a particular *topic string*. A topic string can specify a hierarchy of topics by separating them with a forward slash character (/). This can be visualized by a *topic tree*. For example, if an application publishes to the topic strings `/Sport/American Football` and `/Sport/Soccer`, a topic tree will be created that has a parent node `Sport` with two children, `American Football`, and `Soccer`.

Topics inherit their attributes from the first parent administrative node found in their topic tree. If there are no administrative topic nodes in a particular topic tree, then all topics will inherit their attributes from the base topic object, SYSTEM.BASE.TOPIC.

You can create an administrative topic object at any node in a topic tree by specifying that node's topic string in the TOPICSTR attribute of the administrative topic object. You can also define other attributes for the administrative topic node. For more information about these attributes, see the WebSphere MQ Script (MQSC) Command Reference, or the WebSphere MQ Programmable Command Formats and Administration Interface. Each administrative topic object will, by default, inherit its attributes from its closest parent administrative topic node.

Administrative topic objects can also be used to hide the full topic tree from application developers. If an administrative topic object named `FOOTBALL.US` is created for the topic `/Sport/American Football`, an application can publish or subscribe to the object named `FOOTBALL.US` instead of the string `/Sport/American Football` with the same result.

If you enter a #, +, /, or * character within a topic string on a topic object, the character is treated as a normal character within the string, and is considered to be part of the topic string associated with an administrative topic object.

For more information about administrative topic objects, see the WebSphere MQ Publish/Subscribe User's Guide.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fg19560_

1.1.3.15. System default objects

The *system default objects* are a set of object definitions that are created automatically whenever a queue manager is created. You can copy and modify any of these object definitions for use in applications at your installation.

Default object names have the stem SYSTEM; for example, the default local queue is SYSTEM.DEFAULT.LOCAL.QUEUE, and the default receiver channel is SYSTEM.DEF.RECEIVER. You cannot rename these objects; default objects of these names are required.

When you define an object, any attributes that you do not specify explicitly are copied from the appropriate default object. For example, if you define a local queue, those attributes that you do not specify are taken from the default queue SYSTEM.DEFAULT.LOCAL.QUEUE.

See System and default objects for more information about system defaults.

Parent topic: Objects

This build: January 26, 2011 11:54:28

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa10540_

1.1.4. Clients and servers

WebSphere® MQ supports client-server configurations for its applications.

A WebSphere MQ *client* is a component that allows an application running on a system to issue MQI calls to a queue manager running on another system. The output from the call is sent back to the client, which passes it back to the application.

A WebSphere MQ *server* is a queue manager that provides queuing services to one or more clients. All the WebSphere MQ objects, for example queues, exist only on the queue manager machine (the WebSphere MQ server machine), and not on the client. A WebSphere MQ server can also support local WebSphere MQ applications.

The difference between a WebSphere MQ server and an ordinary queue manager is that a server has a dedicated communications link with each client. For more information about creating channels for clients and servers, see WebSphere MQ Intercommunication.

For information about client support in general, see WebSphere MQ Clients.

WebSphere MQ applications in a client-server environment

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:28

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10550_

1.1.4.1. WebSphere MQ applications in a client-server environment

When linked to a server, client WebSphere® MQ applications can issue most MQI calls in the same way as local applications. The client application issues an **MQCONN** call to connect to a specified queue manager. Any additional MQI calls that specify the connection handle returned from the connect request are then processed by this queue manager.

You must link your applications to the appropriate client libraries. See WebSphere MQ Clients for further information.

Parent topic: Clients and servers

This build: January 26, 2011 11:54:28

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10560_

1.1.5. Extending queue manager facilities

The facilities provided by a queue manager can be extended by:

- User exits
- API exits
- Installable services

See Installable services and components for more information about the installable services.

User exits

API exits

Installable services

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10570_

1.1.5.1. User exits

User exits provide a mechanism for you to insert your own code into a queue manager function. The user exits supported include:

Channel exits
These exits change the way that channels operate. Channel exits are described in WebSphere MQ Intercommunication.

Data conversion exits
These exits create source code fragments that can be put into application programs to convert data from one format to another. Data conversion exits are described in the WebSphere MQ Application Programming Guide.

The cluster workload exit
The function performed by this exit is defined by the provider of the exit. Call definition information is given in WebSphere MQ Queue Manager Clusters.

Parent topic: Extending queue manager facilities

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10580_

1.1.5.2. API exits

API exits let you write code that changes the behavior of WebSphere® MQ API calls, such as MQPUT and MQGET, and then insert that code immediately before or immediately after those calls. The insertion is automatic; the queue manager drives the exit code at the registered points. For more information about API exits, see API exits and the WebSphere MQ Application Programming Guide.

Parent topic: Extending queue manager facilities

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10590_

1.1.5.3. Installable services

Installable services have formalized interfaces (an API) with multiple entry points.

An implementation of an installable service is called a *service component*. You can use the components supplied with WebSphere® MQ, or you can write your own component to perform the functions that you require.

Currently, the following installable services are provided:

Authorization service

The authorization service allows you to build your own security facility.

The default service component that implements the service is the Object Authority Manager (OAM). By default, the OAM is active, and you do not have to do anything to configure it. You can use the authorization service interface to create other components to replace or augment the OAM. For more information about the OAM, see Setting up security on UNIX systems and Windows.

Name service

The name service enables applications to share queues by identifying remote queues as though they were local queues.

You can write your own name service component. You might want to do this if you intend to use the name service with WebSphere MQ, for example. To use the name service you must have either a component that is either user-written, or supplied by a different software vendor. By default, the name service is inactive.

Parent topic: Extending queue manager facilities

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10600_

1.1.6. Security

In WebSphere® MQ, there are three methods of providing security:

- The Object Authority Manager (OAM) facility
- User-written, or third party, channel exits
- Channel security using Secure Sockets Layer (SSL)

Object Authority Manager (OAM) facility

User-written or third party channel exits

Channel security using SSL

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10610_

1.1.6.1. Object Authority Manager (OAM) facility

Authorization for using MQI calls, commands, and access to objects is provided by the **Object Authority Manager** (OAM), which by default is enabled. Access to WebSphere® MQ entities is controlled through WebSphere MQ user groups and the OAM. Administrators can use a command line interface to grant or revoke authorizations as required.

For more information about creating authorization service components, see Setting up security on UNIX systems and Windows.

Parent topic: Security

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10620_

1.1.6.2. User-written or third party channel exits

Channels can use user-written or third party channel exits. For more information, see WebSphere MQ Intercommunication.

Parent topic: Security

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10630_

1.1.6.3. Channel security using SSL

The Secure Sockets Layer (SSL) protocol provides industry-standard channel security, with protection against eavesdropping, tampering, and impersonation.

SSL uses public key and symmetric techniques to provide message privacy and integrity and mutual authentication.

For a comprehensive review of security in WebSphere® MQ including detailed information on SSL, see WebSphere MQ Security. For an overview of SSL, including pointers to the commands described in this book, see Protecting channels with SSL.

Parent topic: Security

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10640_

1.1.7. Transactional support

An application program can group a set of updates into a *unit of work*. These updates are usually logically related and must all be successful for data integrity to be preserved. If one update succeeds while another fails, data integrity is lost.

When a unit of work completes successfully, it is said to *commit*. Once committed, all updates made within that unit of work are made permanent and irreversible. However, if the unit of work fails, all updates are instead *backed out*. This process, where units of work are either committed or backed out with integrity, is known as *sync point coordination*.

A *local* unit of work is one in which the only resources updated are those of the WebSphere® MQ queue manager. Here sync point coordination is provided by the queue manager itself using a single-phase commit process.

A *global* unit of work is one in which resources belonging to other resource managers, such as XA-compliant databases, are also updated. Here, a two-phase commit procedure must be used and the unit of work can be coordinated by the queue manager itself, or externally by another XA-compliant transaction manager such as IBM® TXSeries, or BEA Tuxedo.

For more information, see Transactional support.

WebSphere MQ also provides support for the Microsoft Transaction Server (COM+). Using the Microsoft Transaction Server (COM+) provides information on how to set up WebSphere MQ to take advantage of COM+ support.

Parent topic: Introduction to WebSphere MQ

This build: January 26, 2011 11:54:29

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10650_

1.2. An introduction to WebSphere MQ administration

This chapter introduces WebSphere® MQ administration.

Administration tasks include creating, starting, altering, viewing, stopping, and deleting clusters, processes and WebSphere MQ objects (queue managers, queues, namelists, process definitions, channels, client connection channels, listeners, services, and authentication information objects).

The chapter contains the following sections:

- Local and remote administration
- Performing administration tasks using commands
- Further methods of administration
- Understanding WebSphere MQ file names

Local and remote administration

Performing administration tasks using commands

Further methods of administration

Understanding WebSphere MQ file names

Parent topic: Introduction

This build: January 26, 2011 11:54:29

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10660_

1.2.1. Local and remote administration

You administer WebSphere® MQ objects locally or remotely.

Local administration means carrying out administration tasks on any queue managers you have defined on your local system. You can access other systems, for example through the TCP/IP terminal emulation program **telnet**, and carry out administration there. In WebSphere MQ, you can consider this as local administration because no channels are involved, that is, the communication is managed by the operating system.

WebSphere MQ supports administration from a single point of contact through what is known as *remote administration*. This allows you to issue commands from your local system that are processed on another system. For example, you can issue a remote command to change a queue definition on a remote queue manager. You do not have to log on to that system, although you do need to have the appropriate channels defined. The queue manager and command server on the target system must be running.

Some commands cannot be issued in this way, in particular, creating or starting queue managers and starting command servers. To perform this type of task, you must either log onto the remote system and issue the commands from there or create a process that can issue the commands for you.

Administering remote WebSphere MQ objects describes the subject of remote administration in greater detail.

Parent topic: An introduction to WebSphere MQ administration

This build: January 26, 2011 11:54:29

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10670_

1.2.2. Performing administration tasks using commands

There are three sets of commands that you can use to administer WebSphere® MQ: control commands, MQSC commands, and PCF commands. These command sets are available on all platforms covered by this book.

Control commands

WebSphere MQ Script (MQSC) commands

PCF commands

Parent topic: An introduction to WebSphere MQ administration

This build: January 26, 2011 11:54:30

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10680_

1.2.2.1. Control commands

Control commands allow you to perform administrative tasks on queue managers themselves.

They are described in Managing queue managers.

Parent topic: Performing administration tasks using commands

This build: January 26, 2011 11:54:30

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10690_

1.2.2.2. WebSphere® MQ Script (MQSC) commands

Use MQSC commands to manage queue manager objects, including the queue manager itself, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects.

You issue MQSC commands to a queue manager using the **runmqsc** command. You can do this interactively, issuing commands from a keyboard, or you can redirect the standard input device (stdin) to run a sequence of commands from an ASCII text file. In both cases, the format of the commands is the same.

You can run the **runmqsc** command in three modes, depending on the flags set on the command:

- *Verification mode*, where the MQSC commands are verified on a local queue manager, but are not actually run
- *Direct mode*, where the MQSC commands are run on a local queue manager
- *Indirect mode*, where the MQSC commands are run on a remote queue manager

Object attributes specified in MQSC commands are shown in this book in uppercase (for example, RQMNAME), although they are not case sensitive. MQSC command attribute names are limited to eight characters.

MQSC commands are available on other platforms, including i5/OS®, and z/OS®. MQSC commands are summarized in Comparing command sets.

The WebSphere MQ Script (MQSC) Command Reference contains a description of each MQSC command and its syntax.

See Performing local administration tasks using MQSC commands for more information about using MQSC commands in local administration.

Parent topic: Performing administration tasks using commands

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10700_

1.2.2.3. PCF commands

WebSphere® MQ programmable command format (PCF) commands allow administration tasks to be programmed into an administration program. In this way you can create queues and process definitions, and change queue managers, from a program.

PCF commands cover the same range of functions provided by the MQSC commands. See PCF commands for more information.

You can use the WebSphere MQ Administration Interface (MQAI) to obtain easier programming access to PCF messages. This is described in greater detail in Using the MQAI to simplify the use of PCFs.

Parent topic: Performing administration tasks using commands

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10710_

1.2.3. Further methods of administration

In addition to the available command sets, there are further methods of administration:

- The WebSphere® MQ Explorer
- The Windows Default Configuration application (WebSphere MQ for Windows only)
- Use of the Microsoft Cluster Service (MSCS) (WebSphere MQ for Windows only)

Administration using the WebSphere MQ Explorer
The WebSphere MQ Explorer is an application that runs under the Eclipse platform and is available with WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform). The WebSphere MQ Explorer can be used to administer local or remote queue managers on any supported platform. It provides a graphical user interface for controlling resources in a network.

Using the Windows Default Configuration application

Using the Microsoft Cluster Service (MSCS)

Parent topic: An introduction to WebSphere MQ administration

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10720_

1.2.3.1. Administration using the WebSphere MQ Explorer

The WebSphere® MQ Explorer is an application that runs under the Eclipse platform and is available with WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform). The WebSphere MQ Explorer can be used to administer local or remote queue managers on any supported platform. It provides a graphical user interface for controlling resources in a network.

Using the WebSphere MQ Explorer, you can perform the following actions:

- Define and control various resources including queue managers, queues, process definitions, namelists, channels, client connection channels, listeners, services, and clusters.
- Start or stop a local queue manager and its associated processes.
- View queue managers and their associated objects on your workstation or from other workstations.
- Check the status of queue managers, clusters, and channels.
- Check to see which applications, users, or channels have a particular queue open, from the queue status.

You can invoke the WebSphere MQ Explorer using the **strmqcfg** command, and on Windows from the WebSphere MQ Taskbar application, or from the Windows Start prompt, or from the WebSphere MQ alert monitor application.

On Linux, the WebSphere MQ Explorer might fail to start if you have more than one Eclipse installation. If this happens, start the WebSphere MQ Explorer using a different user ID to the one you use for the other Eclipse installation.

On Linux, to start the WebSphere MQ Explorer successfully, you must be able to write a file to your home directory, and the home directory must exist.

See Administration using the WebSphere MQ Explorer for more information.

Parent topic: Further methods of administration

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa10730_

1.2.3.2. Using the Windows Default Configuration application

You can use the Windows Default Configuration program from the **Prepare WebSphere MQ Wizard** to create a *starter* (or default) set of WebSphere® MQ objects. A summary of the default objects created is listed in Table 1 .

Parent topic: Further methods of administration

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10740_

1.2.3.3. Using the Microsoft Cluster Service (MSCS)

Microsoft Cluster Service (MSCS) enables you to connect servers into a *cluster*, giving higher availability of data and applications, and making it easier to manage the system. MSCS can automatically detect and recover from server or application failures.

It is important not to confuse clusters in the MSCS sense with WebSphere® MQ clusters. The distinction is:

WebSphere MQ clusters
are groups of two or more queue managers on one or more computers, providing automatic interconnection, and allowing queues to be shared amongst them for load balancing and redundancy.

MSCS clusters
are groups of computers, connected together and configured in such a way that, if one fails, MSCS performs a *failover*, transferring the state data of applications from the failing computer to another computer in the cluster and reinitiating their operation there.

Supporting the Microsoft Cluster Service (MSCS) provides detailed information on how to configure your WebSphere MQ for Windows system to use MSCS.
Parent topic: Further methods of administration

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10750_

1.2.4. Understanding WebSphere MQ file names

Each WebSphere® MQ queue manager, queue, process definition, namelist, channel, client connection channel, listener, service, and authentication information object is represented by a file. Because object names are not necessarily valid file names, the queue manager converts the object name into a valid file name where necessary.

The default path to a queue manager directory is as follows:

- A prefix, which is defined in the WebSphere MQ configuration information:
  - On Windows 32-bit systems the default prefix is `C:\Program Files\IBM\WebSphere MQ`. On Windows 64-bit systems the default prefix is, `C:\Program Files\IBM\WebSphere MQ (x86)\` This is configured in the Windows Registry.
  - On UNIX systems the default prefix is `/var/mqm`. This is configured in the `DefaultPrefix` stanza of the mqs.ini configuration file.

  Where available, the prefix can be changed using the WebSphere MQ properties page in the WebSphere MQ Explorer, otherwise edit the `mqs.ini` configuration file manually.
- The queue manager name is transformed into a valid directory name. For example, the queue manager:

  ```
  queue.manager
  ```

  would be represented as:

  ```
  queue!manager
  ```

  This process is referred to as *name transformation*.

  Queue manager name transformation

  Object name transformation

Parent topic: An introduction to WebSphere MQ administration

This build: January 26, 2011 11:54:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10760_

1.2.4.1. Queue manager name transformation

In WebSphere® MQ, you can give a queue manager a name containing up to 48 characters.

For example, you could name a queue manager:

```
QUEUE.MANAGER.ACCOUNTING.SERVICES
```

However, each queue manager is represented by a file and there are limitations on the maximum length of a file name, and on the characters that can be used in the name. As a result, the names of files representing objects are automatically transformed to meet the requirements of the file system.

The rules governing the transformation of a queue manager name are as follows:

1. Transform individual characters:
   - . becomes !

- / becomes &

2. If the name is still not valid:
   a. Truncate it to eight characters
   b. Append a three-character numeric suffix

For example, assuming the default prefix and a queue manager with the name `queue.manager`:

- In WebSphere MQ for Windows with NTFS or FAT32, the queue manager name becomes:
  `c:\Program Files\IBM\WebSphere MQ\qmgrs\queue!manager`
- In WebSphere MQ for Windows with FAT, the queue manager name becomes:
  `c:\Program Files\IBM\WebSphere MQ\qmgrs\queue!ma`
- In WebSphere MQ for UNIX systems, the queue manager name becomes:
  `/var/mqm/qmgrs/queue!manager`

The transformation algorithm also distinguishes between names that differ only in case on file systems that are not case sensitive.

Parent topic: Understanding WebSphere MQ file names

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10770_

1.2.4.2. Object name transformation

Object names are not necessarily valid file system names. You might need to transform your object names. The method used is different from that for queue manager names because, although there are only a few queue manager names on each machine, there can be a large number of other objects for each queue manager. Queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects are represented in the file system.

When a new name is generated by the transformation process, there is no simple relationship with the original object name. You can use the **dspmqfls** command to convert between real and transformed object names.

Parent topic: Understanding WebSphere MQ file names

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10780_

2. Administration using WebSphere® MQ commands

Managing queue managers

Administering local WebSphere MQ objects

Automating administration tasks

Administering remote WebSphere MQ objects

Parent topic: System Administration Guide

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10790_

2.1. Managing queue managers

This chapter tells you how to perform operations on queue managers using control commands and the WebSphere® MQ Explorer.

It contains the following topics:

- Using control commands
- Using the WebSphere MQ Explorer to manage queue managers
- Creating a queue manager
- Starting a queue manager
- Stopping a queue manager
- Restarting a queue manager
- Deleting a queue manager

Using control commands

Using the WebSphere MQ Explorer to manage queue managers
You can use the WebSphere MQ Explorer to perform the operations described in this collection of topics, except for a preemptive shutdown.

Creating a queue manager

Starting a queue manager

Stopping a queue manager

Restarting a queue manager

Deleting a queue manager
You can delete a queue manager using the **dltmqm** command or using the Explorer.

Parent topic: Administration using WebSphere MQ commands

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10800_

2.1.1. Using control commands

Control commands can be divided into three categories, as shown in Table 1.

Table 1. Categories of control commands

Category	Description
Queue manager commands	Queue manager control commands include commands for creating, starting, stopping, and deleting queue managers and command servers
Channel commands	Channel commands include commands for starting and ending channels and channel initiators
Utility commands	Utility commands include commands associated with: • Running MQSC commands • Conversion exits • Authority management • Recording and recovering media images of queue manager resources • Displaying and resolving transactions • Trigger monitors • Displaying the file names of WebSphere® MQ objects • The File Transfer Application

For information about administration tasks for channels, see WebSphere MQ Intercommunication.

Using control commands on Windows systems

Using control commands on UNIX systems

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10810_

2.1.1.1. Using control commands on Windows systems

In WebSphere® MQ for Windows, you enter control commands at a command prompt. In these environments, control commands and their flags are not case sensitive, but arguments to those commands (such as queue names and queue-manager names) are case sensitive.

For example, in the command:

```
crtmqm /u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- The command name can be entered in uppercase or lowercase, or a mixture of the two. These are all valid: `crtmqm`, `CRTMQM`, and `CRTmqm`.
- The flag can be entered as `-u`, `-U`, `/u`, or `/U`.
- `SYSTEM.DEAD.LETTER.QUEUE` and `jupiter.queue.manager` must be entered exactly as shown.

For more information, see How to use WebSphere MQ control commands.

Parent topic: Using control commands

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10820_

2.1.1.2. Using control commands on UNIX systems

In WebSphere® MQ for UNIX systems, you enter control commands in a shell window. In these environments, control commands, including the command name itself, the flags, and any arguments, are case sensitive. For example, in the command:

```
crtmqm -u SYSTEM.DEAD.LETTER.QUEUE jupiter.queue.manager
```

- The command name must be `crtmqm`, not `CRTMQM`.
- The flag must be `-u`, not `-U`.

- The dead-letter queue is called `SYSTEM.DEAD.LETTER.QUEUE`.
- The argument is specified as `jupiter.queue.manager`, which is different from `JUPITER.queue.manager`.

Take care to type the commands exactly as you see them in the examples.

For more information about the **crtmqm** command, see crtmqm.

Parent topic: Using control commands

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10830_

2.1.2. Using the WebSphere MQ Explorer to manage queue managers

You can use the WebSphere® MQ Explorer to perform the operations described in this collection of topics, except for a preemptive shutdown.

The WebSphere MQ Explorer is available on Windows and WebSphere MQ for Linux (x86 platform) systems only.

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10840_

2.1.3. Creating a queue manager

A queue manager manages the resources associated with it, in particular the queues that it owns. It provides queuing services to applications for Message Queuing Interface (MQI) calls and commands to create, modify, display, and delete WebSphere® MQ objects.

Before you can do anything with messages and queues, you must create and start at least one queue manager and its associated objects. To create a queue manager, use the WebSphere MQ control command **crtmqm** (described in crtmqm). The **crtmqm** command automatically creates the required default objects and system objects (described in System default objects). Default objects form the basis of any object definitions that you make; system objects are required for queue manager operation. When you have created a queue manager and its objects, use the **strmqm** command to start the queue manager.

Note: WebSphere MQ does not support machine names that contain spaces. If you install WebSphere MQ on a computer with a machine name that contains spaces, you cannot create any queue managers.

Guidelines for creating queue managers

Creating a default queue manager

Making an existing queue manager the default

Backing up configuration files after creating a queue manager

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10850_

2.1.3.1. Guidelines for creating queue managers

Before you can create a queue manager, there are several points you need to consider (especially in a production environment). Work through the following checklist:

Naming conventions

Use uppercase names so that you can communicate with queue managers on all platforms. Remember that names are assigned exactly as you enter them. To avoid the inconvenience of lots of typing, do not use unnecessarily long names.

Specify a unique queue manager name

When you create a queue manager, ensure that no other queue manager has the same name *anywhere* in your network. Queue manager names are not checked when the queue manager is created, and names that are not unique prevent you from creating channels for distributed queuing.

One way of ensuring uniqueness is to prefix each queue manager name with its own unique node name. For example, if a node is called `ACCOUNTS`, you can name your queue manager `ACCOUNTS.SATURN.QUEUE.MANAGER`, where `SATURN` identifies a particular queue manager and `QUEUE.MANAGER` is an extension you can give to all queue managers. Alternatively, you can omit this, but note that `ACCOUNTS.SATURN` and `ACCOUNTS.SATURN.QUEUE.MANAGER` are *different* queue manager names.

If you are using WebSphere® MQ for communication with other enterprises, you can also include your own enterprise name as a prefix. This is not done in the examples, because it makes them more difficult to follow.

Note: Queue manager names in control commands are case-sensitive. This means that you are allowed to create two queue managers with the names `jupiter.queue.manager` and `JUPITER.queue.manager`. However, it is better to avoid such complications.

Limit the number of queue managers

You can create as many queue managers as resources allow. However, because each queue manager requires its own resources, it is generally better to have one queue manager with 100 queues on a node than to have ten queue managers with ten queues each.

In production systems, many processors can be exploited with a single queue manager, but larger server machines might run more effectively with multiple queue managers.

Specify a default queue manager

Each node should have a default queue manager, though it is possible to configure WebSphere MQ on a node without one. The default queue manager is the queue manager that applications connect to if they do not specify a queue manager name in an **MQCONN** call. It is also the queue manager that processes MQSC commands when you invoke the **runmqsc** command without specifying a queue manager name.

Specifying a queue manager as the default *replaces* any existing default queue manager specification for the node.

Changing the default queue manage can affect other users or applications. The change has no effect on currently-connected applications, because they can use the handle from their original connect call in any further MQI calls. This handle ensures that the calls are directed to the same queue manager. Any applications connecting *after* you have changed the default queue manager connect to the new default queue manager. This might be what you intend, but you should take this into account before you change the default.

Creating a default queue manager is described in Creating a default queue manager.

Specify a dead-letter queue

The dead-letter queue is a local queue where messages are put if they cannot be routed to their intended destination.

It is important to have a dead-letter queue on each queue manager in your network. If you do not define one, errors in application programs might cause channels to be closed, and replies to administration commands might not be received.

For example, if an application tries to put a message on a queue on another queue manager, but gives the wrong queue name, the channel is stopped and the message remains on the transmission queue. Other applications cannot then use this channel for their messages.

The channels are not affected if the queue managers have dead-letter queues. The undelivered message is put on the dead-letter queue at the receiving end, leaving the channel and its transmission queue available.

When you create a queue manager, use the -u flag to specify the name of the dead-letter queue. You can also use an MQSC command to alter the attributes of a queue manager that you have already defined to specify the dead-letter queue to be used. See Altering queue manager attributes for an example of the MQSC command ALTER.

Specify a default transmission queue

A transmission queue is a local queue on which messages in transit to a remote queue manager are queued before transmission. The default transmission queue is the queue that is used when no transmission queue is explicitly defined. Each queue manager can be assigned a default transmission queue.

When you create a queue manager, use the -d flag to specify the name of the default transmission queue. This does not actually create the queue; you have to do this explicitly later on. See Working with local queues for more information.

Specify the logging parameters you require

You can specify logging parameters on the **crtmqm** command, including the type of logging, and the path and size of the log files.

In a development environment, the default logging parameters should be adequate. However, you can change the defaults if, for example:

- You have a low-end system configuration that cannot support large logs.
- You anticipate a large number of long messages being on your queues at the same time.
- You anticipate a lot of persistent messages passing through the queue manager.

Once you have set the logging parameters, some of them can only be changed by deleting the queue manager and recreating it with the same name but with different logging parameters.

For more information about logging parameters, see Availability, recovery and restart.

Parent topic: Creating a queue manager

This build: January 26, 2011 11:54:32

Notices | Trademarks | Downloads | Library | Support | Feedback

Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10860_

2.1.3.2. Creating a default queue manager

You create a default queue manager using the **crtmqm** command with the **-q** flag. The following **crtmqm** command:

- Creates a default queue manager called `SATURN.QUEUE.MANAGER`
- Creates the default and system objects
- Specifies the names of both a default transmission queue and a dead-letter queue

```
crtmqm -q -d MY.DEFAULT.XMIT.QUEUE -u SYSTEM.DEAD.LETTER.QUEUE  SATURN.QUEUE.MANAGER
```

where:

-q
Indicates that this queue manager is the default queue manager.

-d MY.DEFAULT.XMIT.QUEUE
Is the name of the default transmission queue to be used by this queue manager.

Note: WebSphere® MQ does not create a default transmission queue for you; you have to define it yourself.

-u SYSTEM.DEAD.LETTER.QUEUE
Is the name of the default dead-letter queue created by WebSphere MQ on installation.

SATURN.QUEUE.MANAGER
Is the name of this queue manager. This must be the last parameter specified on the **crtmqm** command.

The complete syntax of the **crtmqm** command is shown in crtmqm.

The system and default objects are listed in System and default objects.

For WebSphere MQ for UNIX systems only

You can create the queue manager directory `/var/mqm/qmgrs/<qmgr>`, even on a separate local file system, before you use the **crtmqm** command. When you use **crtmqm**, if the `/var/mqm/qmgrs/<qmgr>` directory exists, is empty, and is owned by mqm, it is used for the queue manager data. If the directory is not owned by mqm, the creation fails with a First Failure Support Technology™ (FFST™) message. If the directory is not empty, a new directory is created.

Parent topic: Creating a queue manager

This build: January 26, 2011 11:54:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10870_

2.1.3.3. Making an existing queue manager the default

You can make an existing queue manager the default queue manager. The way you do this depends on the platform you are using.

WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform) systems

UNIX systems

Parent topic: Creating a queue manager

This build: January 26, 2011 11:54:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10880_

2.1.3.3.1. WebSphere MQ for Windows and WebSphere MQ for Linux (x86 platform) systems

On WebSphere® MQ for Windows and WebSphere MQ for Linux (x86 platform) systems, you can make an existing queue manager the default queue manager as follows:

1. Open the WebSphere MQ Explorer.
2. Right-click **IBM WebSphere MQ**, then select **Properties...**. The Properties for WebSphere MQ panel is displayed.
3. Type the name of the default queue manager into the Default queue manager name field.
4. Click OK.

Parent topic: Making an existing queue manager the default

This build: January 26, 2011 11:54:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10890_

2.1.3.3.2. UNIX systems

When you create a default queue manager, its name is inserted in the `Name` attribute of the `DefaultQueueManager` stanza in the WebSphere® MQ configuration file (mqs.ini). The stanza and its contents are automatically created if they do not exist.

- To make an existing queue manager the default, change the queue manager name on the `Name` attribute to the name of the new default queue manager. You can do this manually, using a text editor.
- If you do not have a default queue manager on the node, and you want to make an existing queue manager the default, create the *`DefaultQueueManager`* stanza with the required name yourself.
- If you accidentally make another queue manager the default and want to revert to the original default queue manager, edit the `DefaultQueueManager` stanza in mqs.ini, replacing the unwanted default queue manager with that of the one you want.

See Configuring WebSphere MQ for information about configuration files.

Parent topic: Making an existing queue manager the default

This build: January 26, 2011 11:54:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10900_

2.1.3.4. Backing up configuration files after creating a queue manager

WebSphere® MQ configuration information is stored in the Registry for Windows systems, and in configuration files on UNIX systems.

Windows and WebSphere MQ for Linux (x86 platform) systems

UNIX systems

Parent topic: Creating a queue manager

This build: January 26, 2011 11:54:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10910_

2.1.3.4.1. Windows and WebSphere MQ for Linux (x86 platform) systems

If you use WebSphere® MQ for Windows, configuration information is stored in the Windows Registry. Use the WebSphere MQ Explorer (see Changing

configuration information on Windows systems) or the **amqmdain** command (see amqmdain) to make changes to the Registry.

If you use WebSphere MQ for Linux (x86 platform), configuration information is stored in configuration files. You can use the WebSphere MQ Explorer (see Changing configuration information on Windows systems).

Parent topic: Backing up configuration files after creating a queue manager

This build: January 26, 2011 11:54:32

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10920_

2.1.3.4.2. UNIX systems

There are two types of configuration file:

- When you install the product, the WebSphere® MQ configuration file (mqs.ini) is created. It contains a list of queue managers that is updated each time you create or delete a queue manager. There is one mqs.ini file per node.
- When you create a new queue manager, a new queue manager configuration file (qm.ini) is automatically created. This contains configuration parameters for the queue manager.

After creating a queue manager, back up your configuration files. Then, if you create another queue manager that causes you problems, you can reinstate the backups when you have removed the source of the problem. As a general rule, back up your configuration files each time you create a new queue manager.

For more information about configuration files, see Configuring WebSphere MQ.

Parent topic: Backing up configuration files after creating a queue manager

This build: January 26, 2011 11:54:32

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10930_

2.1.4. Starting a queue manager

Although you have created a queue manager, it cannot process commands or MQI calls until you start it. You do this using the **strmqm** command as follows:

```
strmqm saturn.queue.manager
```

On WebSphere® MQ for Windows and WebSphere MQ for Linux (x86 platform) systems, you can start a queue manager as follows:

1. Open the WebSphere MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click **Start**. The queue manager starts.

If the queue manager start up takes more than a few seconds WebSphere MQ will issue information messages intermittently detailing the start up progress. For more information on these messages see WebSphere MQ Messages.

The **strmqm** command does not return control until the queue manager has started and is ready to accept connect requests.

Starting a queue manager automatically

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:32

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10940_

2.1.4.1. Starting a queue manager automatically

In WebSphere® MQ for Windows you can start a queue manager automatically when the system starts using the WebSphere MQ Explorer. For more information, see Administration using the WebSphere MQ Explorer.

Parent topic: Starting a queue manager

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10950_

2.1.5. Stopping a queue manager

Use the **endmqm** command to stop a queue manager. For example, to stop a queue manager called `saturn.queue.manager`, type:

```
endmqm saturn.queue.manager
```

On WebSphere® MQ for Windows and WebSphere MQ for Linux (x86 platform) systems, you can stop a queue manager as follows:

1. Open the WebSphere MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click **Stop...**. The End Queue Manager panel is displayed.

4. Select Controlled, or Immediate.
5. Click **OK**. The queue manager stops.

Quiesced shutdown

Immediate shutdown

Preemptive shutdown

If you have problems shutting down a queue manager

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10960_

2.1.5.1. Quiesced shutdown

By default, the **endmqm** command performs a *quiesced* shutdown of the specified queue manager. This might take a while to complete. A quiesced shutdown waits until *all* connected applications have disconnected.

Use this type of shutdown to notify applications to stop. If you issue:

```
endmqm -c saturn.queue.manager
```

you are not told when all applications have stopped. (An `endmqm -c saturn.queue.manager` command is equivalent to an `endmqm saturn.queue.manager` command.)

However, if you issue:

```
endmqm -w saturn.queue.manager
```

the command waits until all applications have stopped and the queue manager has ended.

Parent topic: Stopping a queue manager

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10970_

2.1.5.2. Immediate shutdown

For an *immediate shutdown* any current MQI calls are allowed to complete, but any new calls fail. This type of shutdown does not wait for applications to disconnect from the queue manager.

For an immediate shutdown, type:

```
endmqm -i saturn.queue.manager
```

Parent topic: Stopping a queue manager

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10980_

2.1.5.3. Preemptive shutdown

Attention!

Do not use this method unless all other attempts to stop the queue manager using the **endmqm** command have failed. This method can have unpredictable consequences for connected applications.

If an immediate shutdown does not work, you must resort to a *preemptive* shutdown, specifying the `-p` flag. For example:

```
endmqm -p saturn.queue.manager
```

This stops the queue manager immediately.

For a detailed description of the **endmqm** command and its options, see endmqm.

Parent topic: Stopping a queue manager

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa10990_

2.1.5.4. If you have problems shutting down a queue manager

Problems in shutting down a queue manager are often caused by applications. For example, when applications:

- Do not check MQI return codes properly

- Do not request notification of a quiesce
- Terminate without disconnecting from the queue manager (by issuing an **MQDISC** call)

If a problem occurs when you stop the queue manager, you can break out of the **endmqm** command using Ctrl-C.

You can then issue another **endmqm** command, but this time with a flag that specifies the type of shutdown that you require.

Parent topic: Stopping a queue manager

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11000_

2.1.6. Restarting a queue manager

To restart a queue manager, type:

```
strmqm saturn.queue.manager
```

On WebSphere® MQ for Windows and WebSphere MQ for Linux (x86 platform) systems, you can restart a queue manager is the same way as starting it, as follows:

1. Open the WebSphere MQ Explorer.
2. Select the queue manager from the Navigator View.
3. Click **Start**. The queue manager restarts.

If the queue manager restart takes more than a few seconds WebSphere MQ will issue information messages intermittently detailing the start up progress. For more information on these messages see WebSphere MQ Messages.

Parent topic: Managing queue managers

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11010_

2.1.7. Deleting a queue manager

You can delete a queue manager using the **dltmqm** command or using the Explorer.

> **Before you begin**
> Stop the queue manager by issuing the following command: `dltmqm saturn.queue.manager`

Parent topic: Managing queue managers

Procedure

On WebSphere® MQ for Windows and WebSphere MQ for Linux (x86 platform) systems, you can delete a queue manager as follows:

1. **Open the WebSphere MQ Explorer.**
2. **In the Navigator view, select the queue manager.**
3. **If the queue manager is not stopped, stop it.**
   a. **Right-click the queue manager.**
   b. **Click Stop.**
4. **Right-click the queue manager.**
5. **Click Delete.**

Results

The queue manager is deleted.

Attention:

- **Deleting a queue manager is a drastic step, because you also delete all resources associated with the queue manager, including all queues and their messages and all object definitions. If you use the dltmqm command, there is no displayed prompt that allows you to change your mind; when you press the Enter key all the associated resources are lost.**
- **In WebSphere MQ for Windows, deleting a queue manager also removes the queue manager from the automatic startup list (described in Starting a queue manager automatically). When the command has completed, a `WebSphere MQ queue manager ending` message is displayed; you are not told that the queue manager has been deleted.**
- **Deleting a cluster queue manager does not remove it from the cluster. See the note in the description of dltmqm for more information.**

For a description of the dltmqm command and its options, see dltmqm. Ensure that only trusted administrators have the authority to use this command. (For information about security, see Setting up security on UNIX systems and Windows.)

If this method for deleting a queue manager does not work, see Removing queue managers manually for an alternative.

This build: January 26, 2011 11:54:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11020_

2.2. Administering local WebSphere MQ objects

This chapter tells you how to administer local WebSphere® MQ objects to support application programs that use the Message Queue Interface (MQI). In this context, local administration means creating, displaying, changing, copying, and deleting WebSphere MQ objects. In addition to the approaches detailed in this chapter you can use the WebSphere MQ Explorer to administer local WebSphere MQ objects, see Administration using the WebSphere MQ Explorer.

This chapter contains the following sections:

- Supporting application programs that use the MQI
- Performing local administration tasks using MQSC commands
- Working with queue managers
- Working with local queues
- Working with alias queues
- Working with model queues
- Working with services
- Managing objects for triggering

Supporting application programs that use the MQI

Performing local administration tasks using MQSC commands

Working with queue managers

Working with local queues

Working with alias queues

Working with model queues

Working with services

Managing objects for triggering

Parent topic: Administration using WebSphere MQ commands

This build: January 26, 2011 11:54:33

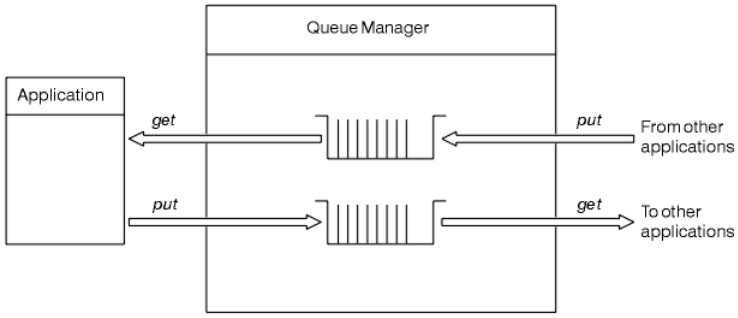
Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11030_

2.2.1. Supporting application programs that use the MQI

WebSphere® MQ application programs need certain objects before they can run successfully. For example, Figure 1 shows an application that removes messages from a queue, processes them, and then sends some results to another queue on the same queue manager.

Figure 1. Queues, messages, and applications



Whereas applications can put messages onto local or remote queues (using MQPUT), they can only get messages directly from local queues (using MQGET).

Before this application can run, the following conditions must be satisfied:

- **The queue manager must exist and be running.**
- **The first application queue, from which the messages are to be removed, must be defined.**
- **The second queue, on which the application puts the messages, must also be defined.**
- **The application must be able to connect to the queue manager. To do this it must be linked to WebSphere MQ. See the WebSphere MQ Application Programming Guide for more information.**
- **The applications that put the messages on the first queue must also connect to a queue manager. If they are remote, they must also be set up with transmission queues and channels. This part of the system is not shown in Figure 1.**

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:34

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11040_

2.2.2. Performing local administration tasks using MQSC commands

This section introduces you to MQSC commands and tells you how to use them for some common tasks. If you use WebSphere® MQ for Windows or WebSphere MQ for Linux (x86 platform), you can also perform the operations described in this section using the WebSphere MQ Explorer. See Administration using the WebSphere MQ Explorer for more information.

You can use MQSC commands to manage queue manager objects, including the queue manager itself, queues, process definitions, channels, client connection channels, listeners, services, namelists, clusters, and authentication information objects. This section deals with queue managers, queues, and process definitions; for information about administering channel, client connection channel, and listener objects, see WebSphere MQ Intercommunication. For information about all the MQSC commands for managing queue manager objects, see WebSphere MQ Script (MQSC) Command Reference.

You issue MQSC commands to a queue manager using the runmqsc command. (For details of this command, see runmqsc.) You can do this interactively, issuing commands from a keyboard, or you can redirect the standard input device (`stdin`) to run a sequence of commands from an ASCII text file. In both cases, the format of the commands is the same. (For information about running the commands from a text file, see Running MQSC commands from text files.)

You can run the runmqsc command in three ways, depending on the flags set on the command:

- **Verify a command without running it, where the MQSC commands are verified on a local queue manager, but are not actually run.**
- **Run a command on a local queue manager, where the MQSC commands are run on a local queue manager.**
- **Run a command on a remote queue manager, where the MQSC commands are run on a remote queue manager.**

You can also run the command followed by a question mark to display the syntax.

Object attributes specified in MQSC commands are shown in this book in uppercase (for example, RQMNAME), although they are not case sensitive. MQSC command attribute names are limited to eight characters. MQSC commands are available on other platforms, including i5/OS® and z/OS®.

MQSC commands are summarized in Comparing command sets.The WebSphere MQ Script (MQSC) Command Reference contains a description of each MQSC command and its syntax.

WebSphere MQ object names

Standard input and output

Using MQSC commands interactively

Running MQSC commands from text files

Running MQSC commands from batch files

Resolving problems with MQSC commands

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:34

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11050_

2.2.2.1. WebSphere® MQ object names

In examples, we use some long names for objects. This is to help you identify the type of object you are dealing with.

When you issue MQSC commands, you need specify only the local name of the queue. In our examples, we use queue names such as:

```
ORANGE.LOCAL.QUEUE
```

The `LOCAL.QUEUE` part of the name is to illustrate that this queue is a local queue. It is *not* required for the names of local queues in general.

We also use the name `saturn.queue.manager` as a queue manager name. The `queue.manager` part of the name is to illustrate that this object is a queue manager. It is *not* required for the names of queue managers in general.

Case-sensitivity in MQSC commands

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11060_

2.2.2.1.1. Case-sensitivity in MQSC commands

MQSC commands, including their attributes, can be written in uppercase or lowercase. Object names in MQSC commands are folded to uppercase (that is, QUEUE and queue are not differentiated), unless the names are enclosed within single quotation marks. If quotation marks are not used, the object is processed with a name in uppercase. See the WebSphere MQ Script (MQSC) Command Reference for more information.

The runmqsc command invocation, in common with all WebSphere® MQ control commands, is case sensitive in some WebSphere MQ environments. See Using control commands for more information.

Parent topic: WebSphere MQ object names

This build: January 26, 2011 11:54:35

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11070_

2.2.2.2. Standard input and output

The *standard input device*, also referred to as `stdin`, is the device from which input to the system is taken. Typically this is the keyboard, but you can specify that input is to come from a serial port or a disk file, for example. The *standard output device*, also referred to as `stdout`, is the device to which output from the system is sent. Typically this is a display, but you can redirect output to a serial port or a file.

On operating-system commands and WebSphere® MQ control commands, the < operator redirects input. If this operator is followed by a file name, input is taken from the file. Similarly, the > operator redirects output; if this operator is followed by a file name, output is directed to that file.

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11080_

2.2.2.3. Using MQSC commands interactively

To use MQSC commands interactively, open a command window or shell and enter:

```
runmqsc
```

In this command, a queue manager name has not been specified, so the MQSC commands are processed by the default queue manager. If you want to use a different queue manager, specify the queue manager name on the runmqsc command. For example, to run MQSC commands on queue manager `jupiter.queue.manager`, use the command:

```
runmqsc jupiter.queue.manager
```

After this, all the MQSC commands you type in are processed by this queue manager, assuming that it is on the same node and is already running.

Now you can type in any MQSC commands, as required. For example, try this one:

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE)
```

For commands that have too many parameters to fit on one line, use continuation characters to indicate that a command is continued on the following line:

- A minus sign (-) indicates that the command is to be continued from the start of the following line.
- A plus sign (+) indicates that the command is to be continued from the first nonblank character on the following line.

Command input terminates with the final character of a nonblank line that is not a continuation character. You can also terminate command input explicitly by entering a semicolon (;). (This is especially useful if you accidentally enter a continuation character at the end of the final line of command input.)

Feedback from MQSC commands

Ending interactive input of MQSC commands

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11090_

2.2.2.3.1. Feedback from MQSC commands

When you issue MQSC commands, the queue manager returns operator messages that confirm your actions or tell you about the errors you have made. For example:

```
AMQ8006: WebSphere MQ queue created.
```

This message confirms that a queue has been created.

```
AMQ8405: Syntax error detected at or near end of command segment below:-

AMQ8426: Valid MQSC commands are:


    ALTER
    CLEAR
    DEFINE
    DELETE
```

```
DISPLAY
END
PING
REFRESH
RESET
RESOLVE
RESUME
START
STOP
SUSPEND
 4 : end
```

This message indicates that you have made a syntax error.

These messages are sent to the standard output device. If you have not entered the command correctly, refer to the WebSphere MQ Script (MQSC) Command Reference for the correct syntax.

Parent topic: Using MQSC commands interactively

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11100_

2.2.2.3.2. Ending interactive input of MQSC commands

To stop working with MQSC commands, enter the END command.

Alternatively, you can use the EOF character for your operating system.

Parent topic: Using MQSC commands interactively

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11110_

2.2.2.4. Running MQSC commands from text files

Running MQSC commands interactively is suitable for quick tests, but if you have very long commands, or are using a particular sequence of commands repeatedly, consider redirecting `stdin` from a text file. (See Standard input and output for information about `stdin` and `stdout`.) To do this, first create a text file containing the MQSC commands using your usual text editor. When you use the runmqsc command, use the redirection operators. For example, the following command runs a sequence of commands contained in the text file `myprog.in`:

```
runmqsc < myprog.in
```

Similarly, you can also redirect the output to a file. A file containing the MQSC commands for input is called an *MQSC command file*. The output file containing replies from the queue manager is called the *output file*.

To redirect both `stdin` and `stdout` on the runmqsc command, use this form of the command:

```
runmqsc < myprog.in > myprog.out
```

This command invokes the MQSC commands contained in the MQSC command file `myprog.in`. Because we have not specified a queue manager name, the MQSC commands run against the default queue manager. The output is sent to the text file `myprog.out`. Figure 1 shows an extract from the MQSC command file `myprog.in` and Figure 1 shows the corresponding extract of the output in `myprog.out`.

To redirect `stdin` and `stdout` on the runmqsc command, for a queue manager (`saturn.queue.manager`) that is not the default, use this form of the command:

```
runmqsc saturn.queue.manager < myprog.in > myprog.out
```

MQSC command files

MQSC command reports

Running the supplied MQSC command files

Using runmqsc to verify commands

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11120_

2.2.2.4.1. MQSC command files

MQSC commands are written in human-readable form, that is, in ASCII text. Figure 1 is an extract from an MQSC command file showing an MQSC command (DEFINE QLOCAL) with its attributes. The WebSphere MQ Script (MQSC) Command Reference contains a description of each MQSC command and its syntax.

Figure 1. Extract from an MQSC command file

```
    .
    .
    .
DEFINE QLOCAL(ORANGE.LOCAL.QUEUE) REPLACE  +
       DESCR(' ') +
       PUT(ENABLED) +
       DEFPRTY(0) +
       DEFPSIST(NO) +
       GET(ENABLED) +
       MAXDEPTH(5000) +
       MAXMSGL(1024) +
       DEFSOPT(SHARED) +
       NOHARDENBO +
       USAGE(NORMAL) +
       NOTRIGGER;
    .
    .
    .
```

For portability among WebSphere® MQ environments, we recommend that you limit the line length in MQSC command files to 72 characters. The plus sign indicates that the command is continued on the next line.

Parent topic: Running MQSC commands from text files

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11130_

2.2.2.4.2. MQSC command reports

The runmqsc command returns a *report*, which is sent to `stdout`. The report contains:

- A header identifying MQSC commands as the source of the report:

  ```
  Starting MQSC for queue manager jupiter.queue.manager.
  ```

  Where `jupiter.queue.manager` is the name of the queue manager.
- An optional numbered listing of the MQSC commands issued. By default, the text of the input is echoed to the output. Within this output, each command is prefixed by a sequence number, as shown in Figure 1. However, you can use the -e flag on the runmqsc command to suppress the output.
- A syntax error message for any commands found to be in error.
- An *operator message* indicating the outcome of running each command. For example, the operator message for the successful completion of a DEFINE QLOCAL command is:

  ```
  AMQ8006: WebSphere MQ queue created.
  ```
- Other messages resulting from general errors when running the script file.
- A brief statistical summary of the report indicating the number of commands read, the number of commands with syntax errors, and the number of commands that could not be processed.
  Note: The queue manager attempts to process only those commands that have no syntax errors.

Figure 1. Extract from an MQSC command report file

```
Starting MQSC for queue manager jupiter.queue.manager.
  .
  .
     12:      DEFINE QLOCAL('ORANGE.LOCAL.QUEUE') REPLACE  +
        :              DESCR(' ') +
```

```
          :              PUT(ENABLED) +
          :              DEFPRTY(0) +
          :              DEFPSIST(NO) +
          :              GET(ENABLED) +
          :              MAXDEPTH(5000) +
          :              MAXMSGL(1024) +
          :              DEFSOPT(SHARED) +
          :              NOHARDENBO +
          :              USAGE(NORMAL) +
          :              NOTRIGGER;
   AMQ8006: WebSphere MQ queue created.
          :
    .
    .
```

Parent topic: Running MQSC commands from text files

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11140_

2.2.2.4.3. Running the supplied MQSC command files

These MQSC command files are supplied with WebSphere® MQ:

amqscos0.tst

Definitions of objects used by sample programs.

amqscic0.tst

Definitions of queues for CICS® transactions.

In WebSphere MQ for Windows, these files are located in the directory `c:\Program Files\IBM\WebSphere MQ\tools\mqsc\samples`.

On UNIX systems these files are located in the directory `opt/mqm/samp` (`usr/mqm/samp` on AIX®).

The command that runs them is:

```
runmqsc < amqscos0.tst >test.out
```

Parent topic: Running MQSC commands from text files

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11150_

2.2.2.4.4. Using runmqsc to verify commands

You can use the runmqsc command to verify MQSC commands on a local queue manager without actually running them. To do this, set the -v flag in the runmqsc command, for example:

```
runmqsc -v < myprog.in > myprog.out
```

When you invoke runmqsc against an MQSC command file, the queue manager verifies each command and returns a report without actually running the MQSC commands. This allows you to check the syntax of the commands in your command file. This is particularly important if you are:

- **Running a large number of commands from a command file.**
- **Using an MQSC command file many times over.**

The returned report is similar to that shown in Figure 1.

You cannot use this method to verify MQSC commands remotely. For example, if you attempt this command:

```
runmqsc -w 30 -v jupiter.queue.manager < myprog.in > myprog.out
```

the -w flag, which you use to indicate that the queue manager is remote, is ignored, and the command is run locally in verification mode. 30 is the number of seconds that WebSphere® MQ waits for replies from the remote queue manager.

Parent topic: Running MQSC commands from text files

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11160_

2.2.2.5. Running MQSC commands from batch files

If you have very long commands, or are using a particular sequence of commands repeatedly, consider redirecting `stdin` from a batch file. To do this, first create a batch file containing the MQSC commands using your usual text editor. When you use the runmqsc command, use the redirection operators. The following example:

1. Creates a test queue manager, TESTQM
2. Creates a matching CLNTCONN and listener set to use TCP/IP port 1600
3. Creates a test queue, TESTQ
4. Puts a message on the queue, using the amqsputc sample program

Figure 1. Example script for running MQSC commands from a batch file

```
export MYTEMPQM=TESTQM
export MYPORT=1600
export MQCHLLIB=/var/mqm/qmgrs/$MQTEMPQM/@ipcc

crtmqm $MYTEMPQM
strmqm $MYTEMPQM
runmqlsr -m $MYTEMPQM -t TCP -p $MYPORT &

runmqsc $MYTEMPQM << EOF
  DEFINE CHANNEL(NTLM) CHLTYPE(SVRCONN) TRPTYPE(TCP)
  DEFINE CHANNEL(NTLM) CHLTYPE(CLNTCONN) QMNAME('$MYTEMPQM') CONNAME('hostname($MYPORT)')
  ALTER  CHANNEL(NTLM) CHLTYPE(CLNTCONN)
  DEFINE QLOCAL(TESTQ)
EOF

amqsputc TESTQ $MYTEMPQM << EOF
hello world
EOF

endmqm -i $MYTEMPQM
```

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11170_

2.2.2.6. Resolving problems with MQSC commands

If you cannot get MQSC commands to run, use the following information to see if any of these common problems apply to you. It is not always obvious what the problem is when you read the error generated.

When you use the runmqsc command, remember the following:

- Use the < operator to redirect input from a file. If you omit this operator, the queue manager interprets the file name as a queue manager name, and issues the following error message:

  ```
  AMQ8118: WebSphere MQ queue manager does not exist.
  ```

- If you redirect output to a file, use the > redirection operator. By default, the file is put in the current working directory at the time runmqsc is invoked. Specify a fully-qualified file name to send your output to a specific file and directory.
- Check that you have created the queue manager that is going to run the commands, by using the following command to display all queue managers:

  ```
  dspmq
  ```

- The queue manager must be running. If it is not, start it; (see Starting a queue manager). You get an error message if you try to start a queue manager that is already running.
- Specify a queue manager name on the runmqsc command if you have not defined a default queue manager, or you get this error:

  ```
  AMQ8146: WebSphere MQ queue manager not available.
  ```

- You cannot specify an MQSC command as a parameter of the runmqsc command. For example, this is not valid:

```
runmqsc DEFINE QLOCAL(FRED)
```

- You cannot enter MQSC commands before you issue the runmqsc command.
- You cannot run control commands from runmqsc. For example, you cannot issue the strmqm command to start a queue manager while you are running MQSC commands interactively. If you do this, you receive error messages similar to the following:

```
runmqsc
  .
  .
Starting MQSC for queue manager jupiter.queue.manager.

     1 : strmqm saturn.queue.manager
AMQ8405: Syntax error detected at or near end of cmd segment below:-s

AMQ8426: Valid MQSC commands are:
     ALTER
     CLEAR
     DEFINE
     DELETE
     DISPLAY
     END
     PING
     REFRESH
     RESET
     RESOLVE
     RESUME
     START
     STOP
     SUSPEND
      2 : end
```

Parent topic: Performing local administration tasks using MQSC commands

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11180_

2.2.3. Working with queue managers

This section contains examples of some MQSC commands that you can use to display or alter queue manager attributes. See the WebSphere MQ Script (MQSC) Command Reference for detailed information about these commands.

Displaying queue manager attributes

Altering queue manager attributes

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11190_

2.2.3.1. Displaying queue manager attributes

To display the attributes of the queue manager specified on the runmqsc command, use the following MQSC command:

```
DISPLAY QMGR
```

Typical output from this command is shown in Figure 1.
Figure 1. Typical output from a DISPLAY QMGR command

```
DISPLAY QMGR
      1 : DISPLAY QMGR
```

```
AMQ8408: Display Queue Manager details.
    QMNAME(SATURN)                                ACCTCONO(DISABLED)
    ACCTINT(1800)                                 ACCTMQI(OFF)
    ACCTQ(OFF)                                    ACTIVREC(MSG)
    ALTDATE(2005-02-09)                           ALTTIME(17.21.40)
    AUTHOREV(DISABLED)                            CCSID(850)
    CHAD(DISABLED)                                CHADEV(DISABLED)
    CHADEXIT( )                                   CHLEV(DISABLED)
    CLWLDATA( )                                   CLWLEXIT( )
    CLWLLEN(100)                                  CLWLMRUC(999999999)
    CLWLUSEQ(LOCAL)                               CMDLEVEL(600)
    COMMANDQ(SYSTEM.ADMIN.COMMAND.QUEUE)          CRDATE(2005-02-09)
    CRTIME(17.21.40)                              DEADQ()
    DEFXMITQ( )                                   DESCR( )
    DISTL(YES)                                    INHIBTEV(DISABLED)
    IPADDRV(IPV4)                                 LOCALEV(DISABLED)
    LOGGEREV(DISABLED)                            MAXHANDS(256)
    MAXMSGL(4194304)                              MAXPRTY(9)
    MAXUMSGS(10000)                               MONACLS(QMGR)
    MONCHL(OFF)                                   MONQ(OFF)
    PERFMEV(DISABLED)                             PLATFORM(WINDOWSNT)
    QMID(SATURN_2005-02-09_02.00.31)              REMOTEEV(DISABLED)
    REPOS( )                                      REPOSNL( )
    ROUTEREC(MSG)                                 SCHINIT(QMGR)
    SCMDSERV(QMGR)                                SSLCRLNL( )
    SSLCRYP( )                                    SSLEV(DISABLED)
    SSLFIPS(NO)
    SSLKEYR(C:\Program Files\IBM\WebSphere MQ\Qmgrs\saturn\ssl\key)
    SSLRKEYC(0)                                   STATACLS(QMGR)
    STATCHL(OFF)                                  STATINT(1800)
    STATMQI(OFF)                                  STATQ(OFF)
    STRSTPEV(ENABLED)                             SYNCPT
    TRIGINT(999999999)
```

The ALL parameter (the default) on the DISPLAY QMGR command displays all the queue manager attributes. In particular, the output tells you the default queue manager name (`saturn.queue.manager`), the dead-letter queue name (SYSTEM.DEAD.LETTER.QUEUE), and the command queue name (SYSTEM.ADMIN.COMMAND.QUEUE).

You can confirm that these queues exist by entering the command:

```
DISPLAY QUEUE (SYSTEM.*)
```

This displays a list of queues that match the stem `SYSTEM.*`. The parentheses are required.

Parent topic: Working with queue managers

This build: January 26, 2011 11:54:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11200_

2.2.3.2. Altering queue manager attributes

To alter the attributes of the queue manager specified on the runmqsc command, use the MQSC command ALTER QMGR, specifying the attributes and values that you want to change. For example, use the following commands to alter the attributes of `jupiter.queue.manager`:

```
runmqsc jupiter.queue.manager

ALTER QMGR DEADQ (ANOTHERDLQ) INHIBTEV (ENABLED)
```

The ALTER QMGR command changes the dead-letter queue used, and enables inhibit events.

Parent topic: Working with queue managers

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11210_

2.2.4. Working with local queues

This section contains examples of some MQSC commands that you can use to manage local, model, and alias queues. See the WebSphere MQ Script (MQSC) Command Reference for detailed information about these commands.

Defining a local queue
For an application, the local queue manager is the queue manager to which the application is connected. Queues managed by the local queue manager are said to be local to that queue manager.

Displaying default object attributes

Copying a local queue definition

Changing local queue attributes

Clearing a local queue

Deleting a local queue

Browsing queues

Monitoring local queues with the Windows Performance Monitor

Enabling large queues

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:36

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11220_

2.2.4.1. Defining a local queue

For an application, the local queue manager is the queue manager to which the application is connected. Queues managed by the local queue manager are said to be local to that queue manager.

Use the MQSC command DEFINE QLOCAL to create a local queue. You can also use the default defined in the default local queue definition, or you can modify the queue characteristics from those of the default local queue.

Note: The default local queue is named SYSTEM.DEFAULT.LOCAL.QUEUE and it was created on system installation.

Using the MQSC command shown below, we define a queue called ORANGE.LOCAL.QUEUE, with the following characteristics:

- It is enabled for gets, enabled for puts, and operates on a priority order basis.
- It is an *normal* queue; it is not an initiation queue or transmission queue, and it does not generate trigger messages.
- The maximum queue depth is 5000 messages; the maximum message length is 4194304 bytes.

```
DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) +
        DESCR('Queue for messages from other systems') +
        PUT (ENABLED) +
        GET (ENABLED) +
        NOTRIGGER +
        MSGDLVSQ (PRIORITY) +
        MAXDEPTH (5000) +
        MAXMSGL (4194304) +
        USAGE (NORMAL);
```

Note:

1. With the exception of the value for the description, all the attribute values shown are the default values. We have shown them here for purposes of illustration. You can omit them if you are sure that the defaults are what you want or have not been changed. See also Displaying default object attributes.
2. USAGE (NORMAL) indicates that this queue is not a transmission queue.
3. If you already have a local queue on the same queue manager with the name ORANGE.LOCAL.QUEUE, this command fails. Use the REPLACE attribute if you want to overwrite the existing definition of a queue, but see also Changing local queue attributes.

Defining a dead-letter queue

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11230_

2.2.4.1.1. Defining a dead-letter queue

Each queue manager should have a local queue to be used as a dead-letter queue so that messages that cannot be delivered to their correct destination can be stored for later retrieval. You must tell the queue manager about the dead-letter queue. You do this by specifying a dead-letter queue name on the crtmqm command (`crtmqm -u DEAD.LETTER.QUEUE`, for example), or by using the DEADQ attribute on the ALTER QMGR command to specify one later. You must define the dead-letter queue before using it.

A sample dead-letter queue called SYSTEM.DEAD.LETTER.QUEUE is available with the product. This queue is automatically created when you create the queue manager. You can modify this definition if required, and rename it.

A dead-letter queue has no special requirements except that:

- It must be a local queue
- Its MAXMSGL (maximum message length) attribute must enable the queue to accommodate the largest messages that the queue manager has to handle plus the size of the dead-letter header (MQDLH)

WebSphere® MQ provides a dead-letter queue handler that allows you to specify how messages found on a dead-letter queue are to be processed or removed. For further information, see The WebSphere MQ dead-letter queue handler.

Parent topic: Defining a local queue

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11240_

2.2.4.2. Displaying default object attributes

When you define a WebSphere® MQ object, it takes any attributes that you do not specify from the default object. For example, when you define a local queue, the queue inherits any attributes that you omit in the definition from the default local queue, which is called SYSTEM.DEFAULT.LOCAL.QUEUE. To see exactly what these attributes are, use the following command:

```
DISPLAY QUEUE (SYSTEM.DEFAULT.LOCAL.QUEUE)
```

The syntax of this command is different from that of the corresponding DEFINE command. On the DISPLAY command you can give just the queue name, whereas on the DEFINE command you have to specify the type of the queue, that is, QLOCAL, QALIAS, QMODEL, or QREMOTE.

You can selectively display attributes by specifying them individually. For example:

```
DISPLAY QUEUE (ORANGE.LOCAL.QUEUE) +
        MAXDEPTH +
        MAXMSGL +
        CURDEPTH;
```

This command displays the three specified attributes as follows:

```
AMQ8409: Display Queue details.
   QUEUE(ORANGE.LOCAL.QUEUE)                TYPE(QLOCAL)
   CURDEPTH(0)                                MAXDEPTH(5000)
   MAXMSGL(4194304)
```

CURDEPTH is the current queue depth, that is, the number of messages on the queue. This is a useful attribute to display, because by monitoring the queue depth you can ensure that the queue does not become full.

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11250_

2.2.4.3. Copying a local queue definition

You can copy a queue definition using the LIKE attribute on the DEFINE command. For example:

```
DEFINE QLOCAL (MAGENTA.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE)
```

This command creates a queue with the same attributes as our original queue ORANGE.LOCAL.QUEUE, rather than those of the system default local queue. Enter the name of the queue to be copied *exactly* as it was entered when you created the queue. If the name contains lower case characters, enclose the name in single quotation marks.

You can also use this form of the DEFINE command to copy a queue definition, but substitute one or more changes to the attributes of the original. For example:

```
DEFINE QLOCAL (THIRD.QUEUE) +
       LIKE (ORANGE.LOCAL.QUEUE) +
       MAXMSGL(1024);
```

This command copies the attributes of the queue ORANGE.LOCAL.QUEUE to the queue THIRD.QUEUE, but specifies that the maximum message length on the new queue is to be 1024 bytes, rather than 4194304.

Note:

1. When you use the LIKE attribute on a DEFINE command, you are copying the queue attributes only. You are not copying the messages on the queue.
2. If you a define a local queue, without specifying LIKE, it is the same as DEFINE LIKE (SYSTEM.DEFAULT.LOCAL.QUEUE).

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11260_

2.2.4.4. Changing local queue attributes

You can change queue attributes in two ways, using either the ALTER QLOCAL command or the DEFINE QLOCAL command with the REPLACE attribute. In Defining a local queue, we defined the queue called ORANGE.LOCAL.QUEUE. Suppose, for example, you want to decrease the maximum message length on this queue to 10 000 bytes.

- Using the ALTER command:

  ```
  ALTER QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000)
  ```

  This command changes a single attribute, that of the maximum message length; all the other attributes remain the same.

- Using the DEFINE command with the REPLACE option, for example:

  ```
  DEFINE QLOCAL (ORANGE.LOCAL.QUEUE) MAXMSGL(10000) REPLACE
  ```

  This command changes not only the maximum message length, but also all the other attributes, which are given their default values. The queue is now put enabled whereas previously it was put inhibited. Put enabled is the default, as specified by the queue SYSTEM.DEFAULT.LOCAL.QUEUE.
  If you *decrease* the maximum message length on an existing queue, existing messages are not affected. Any new messages, however, must meet the new criteria.

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11270_

2.2.4.5. Clearing a local queue

To delete all the messages from a local queue called MAGENTA.QUEUE, use the following command:

```
CLEAR QLOCAL (MAGENTA.QUEUE)
```

Note: There is no prompt that enables you to change your mind; once you press the Enter key the messages are lost.

You cannot clear a queue if:

- **There are uncommitted messages that have been put on the queue under sync point.**
- **An application currently has the queue open.**

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11280_

2.2.4.6. Deleting a local queue

Use the MQSC command DELETE QLOCAL to delete a local queue. A queue cannot be deleted if it has uncommitted messages on it. However, if the queue has one or more committed messages and no uncommitted messages, it can be deleted only if you specify the PURGE option. For example:

```
DELETE QLOCAL (PINK.QUEUE) PURGE
```

Specifying NOPURGE instead of PURGE ensures that the queue is not deleted if it contains any committed messages.

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11290_

2.2.4.7. Browsing queues

WebSphere® MQ provides a sample queue browser that you can use to look at the contents of the messages on a queue. The browser is supplied in both source and executable formats.

In WebSphere MQ for Windows, the default file names and paths are:

Source

`c:\Program Files\IBM\WebSphere MQ\tools\c\samples\`

Executable

`c:\Program files\IBM\WebSphere MQ\tools\c\samples\bin\amqsbcg.exe`

In WebSphere MQ for UNIX, the default file names and paths are:

Source

`/opt/mqm/samp/amqsbcg0.c` **(**`/usr/mqm/samp/amqsbcg0.c` **on AIX®)**

Executable

`/opt/mqm/samp/bin/amqsbcg` **(**`/usr/mqm/samp/bin/amqsbcg` **on AIX)**

The sample requires two input parameters, the queue name and the queue manager name. For example:

```
amqsbcg SYSTEM.ADMIN.QMGREVENT.tpp01 saturn.queue.manager
```

Typical results from this command are shown in Figure 1.

Figure 1. Typical results from queue browser

```
AMQSBCG0 - starts here
**********************

 MQOPEN - 'SYSTEM.ADMIN.QMGR.EVENT'


 MQGET of message number 1
****Message descriptor****

  StrucId  : 'MD  '  Version : 2
```

```
    Report    : 0  MsgType : 8
    Expiry    : -1  Feedback : 0
    Encoding : 546  CodedCharSetId : 850
    Format : 'MQEVENT '
    Priority : 0  Persistence : 0
    MsgId : X'414D512073617475726E2E71756575650005D30033563DB8'
    CorrelId : X'000000000000000000000000000000000000000000000000'
    BackoutCount : 0
    ReplyToQ        : '                                                    '
    ReplyToQMgr     : 'saturn.queue.manager                               '
    ** Identity Context
    UserIdentifier : '             '
    AccountingToken :
     X'0000000000000000000000000000000000000000000000000000000000000000'
    ApplIdentityData : '                                        '
    ** Origin Context
    PutApplType     : '7'
    PutApplName     : 'saturn.queue.manager         '
    PutDate   : '19970417'     PutTime   : '15115208'
    ApplOriginData : '     '

    GroupId : X'000000000000000000000000000000000000000000000000'
    MsgSeqNumber    : '1'
    Offset          : '0'
    MsgFlags        : '0'
    OriginalLength : '104'

****    Message       ****

 length - 104 bytes

00000000:  0700 0000 2400 0000 0100 0000 2C00 0000 '....→.......,...'
00000010:  0100 0000 0100 0000 0100 0000 AE08 0000 '................'
00000020:  0100 0000 0400 0000 4400 0000 DF07 0000 '........D.......'
00000030:  0000 0000 3000 0000 7361 7475 726E 2E71 '....0...saturn.q'
00000040:  7565 7565 2E6D 616E 6167 6572 2020 2020 'ueue.manager     '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020 '                '
00000060:  2020 2020 2020 2020                     '                '

 No more messages
 MQCLOSE
 MQDISC
```

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11300_

2.2.4.8. Monitoring local queues with the Windows Performance Monitor

Administrators of WebSphere® MQ for Windows can monitor the performance of local queues using the Windows Performance Monitor. For more information on the Windows Performance Monitor see the Monitoring WebSphere MQ book.

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11310_

2.2.4.9. Enabling large queues

WebSphere® MQ supports queues larger than 2 GB. On Windows systems, support for large files is available without any additional enablement. On AIX®, HP-UX, Linux, and Solaris systems, you need to explicitly enable large file support before you can create queue files larger than 2 GB. See your operating system documentation for

information on how to do this.

Some utilities, such as tar, cannot cope with files greater than 2 GB. Before enabling large file support, check your operating system documentation for information on restrictions on utilities you use.

Parent topic: Working with local queues

This build: January 26, 2011 11:54:36

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11320_

2.2.5. Working with alias queues

An alias queue provides a way of referring indirectly to another queue. The other queue can be either:

- A local queue (see Defining a local queue).
- A local definition of a remote queue (see Creating a local definition of a remote queue).
- A topic.

An alias queue is not a real queue, but a definition that resolves to a real (or target) queue at run time. The alias queue definition specifies the target queue. When an application makes an MQOPEN call to an alias queue, the queue manager resolves the alias to the target queue name. An alias queue cannot resolve to another alias queue.

Alias queues are useful for:

- Giving different applications different levels of access authorities to the target queue.
- Allowing different applications to work with the same queue in different ways. (Perhaps you want to assign different default priorities or different default persistence values.)
- Simplifying maintenance, migration, and workload balancing. (Perhaps you want to change the target queue name without having to change your application, which continues to use the alias.)

For example, assume that an application has been developed to put messages on a queue called MY.ALIAS.QUEUE. It specifies the name of this queue when it makes an MQOPEN request and, indirectly, if it puts a message on this queue. The application is not aware that the queue is an alias queue. For each MQI call using this alias, the queue manager resolves the real queue name, which could be either a local queue or a remote queue defined at this queue manager.

By changing the value of the TARGQ attribute, you can redirect MQI calls to another queue, possibly on another queue manager. This is useful for maintenance, migration, and load-balancing.

Defining an alias queue

Using other commands with alias queues

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:36

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11330_

2.2.5.1. Defining an alias queue

The following command creates an alias queue:

```
DEFINE QALIAS (MY.ALIAS.QUEUE) TARGET (YELLOW.QUEUE)
```

This command redirects MQI calls that specify MY.ALIAS.QUEUE to the queue YELLOW.QUEUE. The command does not create the target queue; the MQI calls fail if the queue YELLOW.QUEUE does not exist at run time.

If you change the alias definition, you can redirect the MQI calls to another queue. For example:

```
ALTER QALIAS (MY.ALIAS.QUEUE) TARGQ (MAGENTA.QUEUE)
```

This command redirects MQI calls to another queue, MAGENTA.QUEUE.

You can also use alias queues to make a single queue (the target queue) appear to have different attributes for different applications. You do this by defining two aliases, one for each application. Suppose there are two

applications:

- Application ALPHA can put messages on YELLOW.QUEUE, but is not allowed to get messages from it.
- Application BETA can get messages from YELLOW.QUEUE, but is not allowed to put messages on it.

The following command defines an alias that is put enabled and get disabled for application ALPHA:

```
DEFINE QALIAS (ALPHAS.ALIAS.QUEUE) +
        TARGQ (YELLOW.QUEUE) +
        PUT (ENABLED) +
        GET (DISABLED)
```

The following command defines an alias that is put disabled and get enabled for application BETA:

```
DEFINE QALIAS (BETAS.ALIAS.QUEUE) +
        TARGQ (YELLOW.QUEUE) +
        PUT (DISABLED) +
        GET (ENABLED)
```

ALPHA uses the queue name ALPHAS.ALIAS.QUEUE in its MQI calls; BETA uses the queue name BETAS.ALIAS.QUEUE. They both access the same queue, but in different ways.

You can use the LIKE and REPLACE attributes when you define queue aliases, in the same way that you use these attributes with local queues.

Parent topic: Working with alias queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11340_

2.2.5.2. Using other commands with alias queues

You can use the appropriate MQSC commands to display or alter alias queue attributes, or to delete the alias queue object. For example:

Use the following command to display the alias queue's attributes:

```
DISPLAY QUEUE (ALPHAS.ALIAS.QUEUE)
```

Use the following command to alter the base queue name, to which the alias resolves, where the `force` option forces the change even if the queue is open:

```
ALTER QALIAS (ALPHAS.ALIAS.QUEUE) TARGQ(ORANGE.LOCAL.QUEUE) FORCE
```

Use the following command to delete this queue alias:

```
DELETE QALIAS (ALPHAS.ALIAS.QUEUE)
```

You cannot delete an alias queue if an application currently has the queue open. See the WebSphere MQ Script (MQSC) Command Reference for more information about this and other alias queue commands.

Parent topic: Working with alias queues

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11350_

2.2.6. Working with model queues

A queue manager creates a *dynamic queue* if it receives an MQI call from an application specifying a queue name that has been defined as a model queue. The name of the new dynamic queue is generated by the queue manager when the queue is created. A *model queue* is a template that specifies the attributes of any dynamic queues created from it.

Model queues provide a convenient method for applications to create queues as required.

Defining a model queue

Using other commands with model queues

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11360_

2.2.6.1. Defining a model queue

You define a model queue with a set of attributes in the same way that you define a local queue. Model queues and local queues have the same set of attributes, except that on model queues you can specify whether the dynamic queues created are temporary or permanent. (Permanent queues are maintained across queue manager restarts, temporary ones are not.) For example:

```
DEFINE QMODEL (GREEN.MODEL.QUEUE) +
        DESCR('Queue for messages from application X') +
        PUT (DISABLED) +
        GET (ENABLED) +
        NOTRIGGER +
        MSGDLVSQ (FIFO) +
        MAXDEPTH (1000) +
        MAXMSGL (2000) +
        USAGE (NORMAL) +
        DEFTYPE (PERMDYN)
```

This command creates a model queue definition. From the DEFTYPE attribute, you can see that the actual queues created from this template are permanent dynamic queues. Any attributes not specified are automatically copied from the SYSYTEM.DEFAULT.MODEL.QUEUE default queue.

You can use the LIKE and REPLACE attributes when you define model queues, in the same way that you use them with local queues.

Parent topic: Working with model queues

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11370_

2.2.6.2. Using other commands with model queues

You can use the appropriate MQSC commands to display or alter a model queue's attributes, or to delete the model queue object. For example:

Use the following command to display the model queue's attributes:

```
DISPLAY QUEUE (GREEN.MODEL.QUEUE)
```

Use the following command to alter the model to enable puts on any dynamic queue created from this model:

```
ALTER QMODEL (BLUE.MODEL.QUEUE) PUT(ENABLED)
```

Use the following command to delete this model queue:

```
DELETE QMODEL (RED.MODEL.QUEUE)
```

Parent topic: Working with model queues

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11380_

2.2.7. Working with services

Service objects are a means by which additional processes can be managed as part of a queue manager. With services you can define programs that are started and stopped when the queue manager starts and ends.

Service objects can be either of the following types:

Server

A server is a service object that has the parameter SERVTYPE specified as SERVER. A server service object is the definition of a program that will be executed when a specified queue manager is started. Server service objects define programs that typically run for a long period of time. For example, a server service object can be used to execute a trigger monitor process, such as runmqtrm.

Only one instance of a server service object can run concurrently. The status of running server service objects can be monitored using the MQSC command, DISPLAY SVSTATUS.

Command

A command is a service object that has the parameter SERVTYPE specified as COMMAND. Command service objects are similar to server service objects, however multiple instances of a command service object can run concurrently and their status can not be monitored using the MQSC command DISPLAY SVSTATUS.

If the MQSC command, STOP SERVICE, is executed no check is made to determine whether the program started by the MQSC command, START SERVICE, is still active before executing the stop program.

Defining a service object

Managing services

Additional environment variables

Replaceable inserts on service definitions

Examples on using service objects

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11390_

2.2.7.1. Defining a service object

The attributes used to define a service object are:

SERVTYPE

Defines the type of the service object. Possible values are:

SERVER

A server service object.

Only one instance of a server service object can be executed at a time. The status of server service objects can be monitored using the MQSC command, DISPLAY SVSTATUS.

COMMAND

A command service object.

Multiple instances of a command service object can be executed concurrently. The status of a command service objects cannot be monitored.

STARTCMD

The program that is executed to start the service. A fully qualified path to the program must be specified.

STARTARG

Arguments passed to the start program.

STDERR

Specifies the path to a file to which the standard error (stderr) of the service program should be redirected.

STDOUT

Specifies the path to a file to which the standard output (stdout) of the service program should be redirected.

STOPCMD

The program that is executed to stop the service. A fully qualified path to the program must be specified.

STOPARG

Arguments passed to the stop program.

CONTROL

Specifies how the service is to be started and stopped:

MANUAL

The service is not to be started automatically or stopped automatically. It is controlled by use of the START SERVICE and STOP SERVICE commands. This is the default value.

QMGR

The service being defined is to be started and stopped at the same time as the queue manager is started and stopped.

STARTONLY

The service is to be started at the same time as the queue manager is started, but is not requested to stop when the queue manager is stopped.

Parent topic: Working with services

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11400_

2.2.7.2. Managing services

By using the CONTROL parameter, an instance of a service object can be either started and stopped automatically by the queue manager, or started and stopped using the MQSC commands START SERVICE and STOP SERVICE.

When an instance of a service object is started, a message is written to the queue manager error log containing the name of the service object and the process id of the started process. An example log entry for a server service object starting follows:

```
02/15/2005 11:54:24 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
AMQ5028: The Server 'S1' has started. ProcessId(13031).

EXPLANATION:
The Server process has started.
ACTION:
None.
```

An example log entry for a command service object starting follows:

```
02/15/2005 11:53:55 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
AMQ5030: The Command 'C1' has started. ProcessId(13030).

EXPLANATION:
The Command has started.
ACTION:
None.
```

When an instance server service stops, a message is written to the queue manager error logs containing the name of the service and the process id of the ending process. An example log entry for a server service object stopping follows:

```
02/15/2005 11:54:54 AM - Process(10363.1) User(mqm) Program(amqzmgr0)
AMQ5029: The Server 'S1' has ended. ProcessId(13031).

EXPLANATION:
The Server process has ended.
ACTION:
None.
```

Parent topic: Working with services

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11410_

2.2.7.3. Additional environment variables

When a service is started, the environment in which the service process is started is inherited from the environment of the queue manager. It is possible to define additional environment variables to be set in the environment of the service process by adding the variables you want to define to one of the `service.env` environment override files.

There are two possible files to which you can add environment variables:

- The machine scope `service.env` file, which is located in `/var/mqm` on UNIX systems, or in the data directory selected during installation on Windows systems.
- The queue manager scope `service.env` file, which is located in the queue manager data directory. For example, the location of the environment override file for a queue manager named `QMNAME` is:
  - On UNIX systems, `/var/mqm/qmgrs/QMNAME/service.env`.
  - On Windows systems, `C:\Program Files\IBM\WebSphere MQ\qmgrs\QMNAME\service.env`.

Both files are processed, if available, with definitions in the queue manager scope file taking precedence over those in the machine scope file.

The format of the variables defined in the file, service.env, is a list of name and value variable pairs. Each variable must be defined on a new line, and each variable is taken as it is explicitly defined, including white space. An example of the file, service.env, follows:

```
#********************************************************************#
#*                                                                  *#
#* <N_OCO_COPYRIGHT>                                              *#
#* Licensed Materials - Property of IBM                           *#
#*                                                                  *#
#* 63H9336                                                        *#
#* (C) Copyright IBM Corporation 2005                             *#
#*                                                                  *#
#* <NOC_COPYRIGHT>                                                *#
#*                                                                  *#
#********************************************************************#
#********************************************************************#
#* Module Name: service.env                                         *#
#* Type       : WebSphere MQ service environment file               *#
#* Function   : Define additional environment variables to be set   *#
#*              for SERVICE programs.                               *#
#* Usage      : <VARIABLE>=<VALUE>                                  *#
#*                                                                    *#
#********************************************************************#
MYLOC=/opt/myloc/bin
MYTMP=/tmp
TRACEDIR=/tmp/trace
MYINITQ=ACCOUNTS.INITIATION.QUEUE
```

Parent topic: Working with services

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11420_

2.2.7.4. Replaceable inserts on service definitions

In the definition of a service object, it is possible to substitute tokens. Tokens that are substituted will automatically be replaced with their expanded text when the service program is executed. Substitute tokens can be taken from the following list of common tokens, or from any variables that are defined in the file, service.env.

Common tokens

Parent topic: Working with services

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11430_

2.2.7.4.1. Common tokens

The following are common tokens that can be used to substitute tokens in the definition of a service object:

MQ_INSTALL_PATH

The install location of WebSphere® MQ:

- On AIX® systems, the install location is `/usr/mqm/`
- On Solaris, HP-UX, or Linux systems, the install location is `/opt/mqm/`
- On Windows systems, the install location is the install directory selected during the installation of WebSphere MQ

MQ_DATA_PATH

The location of the WebSphere MQ data directory:

- On UNIX systems, the WebSphere MQ data directory location is `/var/mqm/`
- On Windows systems, the location of the WebSphere MQ data directory is the data directory selected during the installation of WebSphere MQ

QMNAME

The current queue manager name.

MQ_SERVICE_NAME

The name of the service.

MQ_SERVER_PID

This token can only be used by the STOPARG and STOPCMD arguments.

For server service objects this token is replaced with the process id of the process started by the STARTCMD and STARTARG arguments. Otherwise, this token is replaced with 0.

MQ_Q_MGR_DATA_PATH

The location of the queue manager data directory.

MQ_Q_MGR_DATA_NAME

The transformed name of the queue manager. For more information on name transformation, see Understanding WebSphere MQ file names.

To use replaceable inserts, insert the token within + characters into any of the STARTCMD, STARTARG, STOPCMD, STOPARG, STDOUT or STDERR strings. For examples of this, see Examples on using service objects.

Parent topic: Replaceable inserts on service definitions

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11440_

2.2.7.5. Examples on using service objects

The following services are written with UNIX style path separator characters, except where otherwise stated.

Using a server service object

Using a command service object

Using a command service object when a queue manager ends only

More on passing arguments

Autostarting a Service

Parent topic: Working with services

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11450_

2.2.7.5.1. Using a server service object

This example shows how to define, use, and alter, a server service object to start a trigger monitor.

1. A server service object is defined, using the following MQSC command:

```
DEFINE SERVICE(S1) +
        CONTROL(QMGR) +
        SERVTYPE(SERVER) +
        STARTCMD('+MQ_INSTALL_PATH+ bin/runmqtrm') +
        STARTARG('-m +QMNAME+ -q ACCOUNTS.INITIATION.QUEUE') +
        STOPCMD('+MQ_INSTALL_PATH+ bin/amqsstop') +
        STOPARG('-m +QMNAME+ -p +MQ_SERVER_PID+')
```

   Where:

   `+MQ_INSTALL_PATH+` is a token representing the installation directory.

   `+QMNAME+` is a token representing the name of the queue manager.

   `ACCOUNTS.INITIATION.QUEUE` is the initiation queue.

   `amqsstop` is a sample program provided with WebSphere® MQ which requests the queue manager to break all connections for the process id. `amqsstop` generates PCF commands, therefore the command server must be running.

   `+MQ_SERVER_PID+` is a token representing the process id passed to the stop program.

2. An instance of the server service object will execute when the queue manager is next started. However, we will start an instance of the server service object immediately with the following MQSC command:

```
START SERVICE(S1)
```

3. The status of the server service process is displayed, using the following MQSC command:

```
DISPLAY SVSTATUS(S1)
```

4. This example now shows how to alter the server service object and have the updates picked up by manually restarting the server service process. The server service object is altered so that the initiation queue is specified as `JUPITER.INITIATION.QUEUE`. The following MQSC command is used:

```
ALTER SERVICE(S1) +
      STARTARG('-m +QMNAME+ -q JUPITER.INITIATION.QUEUE')
```

   Note: A running service will not pick up any updates to its service definition until it is restarted.

5. The server service process is restarted so that the alteration is picked up, using the following MQSC commands:

```
STOP SERVICE(S1)
```

   Followed by:

```
START SERVICE(S1)
```

   The server service process is restarted and picks up the alterations made in 4.
   Note: The MQSC command, `STOP SERVICE`, can only be used if a STOPCMD argument is specified in the service definition.

Parent topic: Examples on using service objects

This build: January 26, 2011 11:54:37

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11460_

2.2.7.5.2. Using a command service object

This example shows how to define a command service object to start a program that writes entries to the operating system's system log when a queue manager is started or stopped:

1. The command service object is defined, using the following MQSC command:

```
DEFINE SERVICE(S2) +
        CONTROL(QMGR) +
        SERVTYPE(COMMAND) +
        STARTCMD('/usr/bin/logger') +
        STARTARG('Queue manager +QMNAME+ starting') +
        STOPCMD('/usr/bin/logger') +
        STOPARG('Queue manager +QMNAME+ stopping')
```

   Where:

   `logger` is the UNIX supplied command to write to the system log.

   `+QMNAME+` is a token representing the name of the queue manager.

Parent topic: Examples on using service objects

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11470_

2.2.7.5.3. Using a command service object when a queue manager ends only

This example shows how to define a command service object to start a program that writes entries to the operating system's system log when a queue manager is stopped only:

1. **The command service object is defined, using the following MQSC command:**

```
DEFINE SERVICE(S3) +
       CONTROL(QMGR) +
       SERVTYPE(COMMAND) +
       STOPCMD('/usr/bin/logger') +
       STOPARG('Queue manager +QMNAME+ stopping')
```

Where:

`logger` **is a sample program provided with WebSphere® MQ that can write entries to the operating system's system log.**

`+QMNAME+` **is a token representing the name of the queue manager.**

Parent topic: Examples on using service objects

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11480_

2.2.7.5.4. More on passing arguments

This example is written with Windows style path separator characters.

This example shows how to define a server service object to start a program called `runserv` **when a queue manager is started. One of the arguments that is to be passed to the starting program is a string containing a space. This argument needs to be passed as a single string. To achieve this, double quotes are used as shown in the following command to define the command service object:**

1. **The server service object is defined, using the following MQSC command:**

```
DEFINE SERVICE(S1) SERVTYPE(SERVER) CONTROL(QMGR) +
  STARTCMD('C:\Program Files\Tools\runserv.exe') +
  STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
  STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')


DEFINE SERVICE(S4) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('C:\Program Files\Tools\runserv.exe') +
       STARTARG('-m +QMNAME+ -d "C:\Program Files\Tools\"') +
       STDOUT('C:\Program Files\Tools\+MQ_SERVICE_NAME+.out')
```

Where:

`+QMNAME+` **is a token representing the name of the queue manager.**

`"C:\Program Files\Tools\"` **is a string containing a space, which will be passed as a single string.**

Parent topic: Examples on using service objects

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11490_

2.2.7.5.5. Autostarting a Service

This example shows how to define a server service object that can be used to automatically start the Trigger Monitor when the queue manager starts.

1. The server service object is defined, using the following MQSC command:

```
DEFINE SERVICE(TRIG_MON_START) +
       CONTROL(QMGR) +
       SERVTYPE(SERVER) +
       STARTCMD('runmqtrm') +
       STARTARG('-m +QMNAME+ -q +IQNAME+')
```

Where:

+QMNAME+ is a token representing the name of the queue manager.

+IQNAME+ is an environment variable defined by the user in one of the service.env files representing the name of the initiation queue.

Parent topic: Examples on using service objects

This build: January 26, 2011 11:54:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11500_

2.2.8. Managing objects for triggering

WebSphere® MQ enables you to start an application automatically when certain conditions on a queue are met. For example, you might want to start an application when the number of messages on a queue reaches a specified number. This facility is called *triggering* and is described in detail in the WebSphere MQ Application Programming Guide.

This section tells you how to set up the required objects to support triggering on WebSphere MQ.

Defining an application queue for triggering

Defining an initiation queue

Defining a process

Displaying attributes of a process definition

Parent topic: Administering local WebSphere MQ objects

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11510_

2.2.8.1. Defining an application queue for triggering

An application queue is a local queue that is used by applications for messaging, through the MQI. Triggering requires a number of queue attributes to be defined on the application queue. Triggering itself is enabled by the *Trigger* attribute (TRIGGER in MQSC commands).

In this example, a trigger event is to be generated when there are 100 messages of priority 5 or greater on the local queue MOTOR.INSURANCE.QUEUE, as follows:

```
DEFINE QLOCAL (MOTOR.INSURANCE.QUEUE) +
       PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
       MAXMSGL (2000) +
       DEFPSIST (YES) +
       INITQ (MOTOR.INS.INIT.QUEUE) +
       TRIGGER +
       TRIGTYPE (DEPTH) +
       TRIGDPTH (100)+
       TRIGMPRI (5)
```

where:

`QLOCAL (MOTOR.INSURANCE.QUEUE)`

Is the name of the application queue being defined.

`PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)`

Is the name of the process definition that defines the application to be started by a trigger monitor program.

`MAXMSGL (2000)`

Is the maximum length of messages on the queue.

`DEFPSIST (YES)`

Specifies that messages on this queue are persistent by default.

`INITQ (MOTOR.INS.INIT.QUEUE)`

Is the name of the initiation queue on which the queue manager is to put the trigger message.

`TRIGGER`

Is the trigger attribute value.

`TRIGTYPE (DEPTH)`

Specifies that a trigger event is generated when the number of messages of the required priority (TRIGMPRI) reaches the number specified in TRIGDPTH.

`TRIGDPTH (100)`

Is the number of messages required to generate a trigger event.

`TRIGMPRI (5)`

Is the priority of messages that are to be counted by the queue manager in deciding whether to generate a trigger event. Only messages with priority 5 or higher are counted.

Parent topic: Managing objects for triggering

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11520_

2.2.8.2. Defining an initiation queue

When a trigger event occurs, the queue manager puts a trigger message on the initiation queue specified in the application queue definition. Initiation queues have no special settings, but you can use the following definition of the local queue MOTOR.INS.INIT.QUEUE for guidance:

```
DEFINE QLOCAL(MOTOR.INS.INIT.QUEUE) +
       GET (ENABLED) +
       NOSHARE +
       NOTRIGGER +
       MAXMSGL (2000) +
       MAXDEPTH (1000)
```

Parent topic: Managing objects for triggering

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11530_

2.2.8.3. Defining a process

Use the DEFINE PROCESS command to create a process definition. A process definition defines the application to be used to process messages from the application queue. The application queue definition names the process to be used and thereby associates the application queue with the application to be used to process its messages. This is done through the PROCESS attribute on the application queue MOTOR.INSURANCE.QUEUE. The following MQSC command defines the required process, MOTOR.INSURANCE.QUOTE.PROCESS, identified in this example:

```
DEFINE PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) +
               DESCR ('Insurance request message processing') +
               APPLTYPE (UNIX) +
               APPLICID ('/u/admin/test/IRMP01') +
               USERDATA ('open, close, 235')
```

Where:

`MOTOR.INSURANCE.QUOTE.PROCESS`

Is the name of the process definition.

`DESCR ('Insurance request message processing')`

Describes the application program to which this definition relates. This text is displayed when you use the DISPLAY PROCESS command. This can help you to identify what the process does. If you use spaces in the string, you must enclose the string in single quotation marks.

`APPLTYPE (UNIX)`

Is the type of application to be started.

`APPLICID ('/u/admin/test/IRMP01')`

Is the name of the application executable file, specified as a fully qualified file name. In Windows systems, a typical `APPLICID` value would be `c:\appl\test\irmp01.exe`.

`USERDATA ('open, close, 235')`

Is user-defined data, which can be used by the application.

Parent topic: Managing objects for triggering

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11540_

2.2.8.4. Displaying attributes of a process definition

Use the DISPLAY PROCESS command to examine the results of your definition. For example:

```
DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)


     24 : DISPLAY PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS) ALL
AMQ8407: Display Process details.
     DESCR ('Insurance request message processing')
     APPLICID ('/u/admin/test/IRMP01')
     USERDATA (open, close, 235)
     PROCESS (MOTOR.INSURANCE.QUOTE.PROCESS)
     APPLTYPE (UNIX)
```

You can also use the MQSC command ALTER PROCESS to alter an existing process definition, and the DELETE PROCESS command to delete a process definition.

Parent topic: Managing objects for triggering

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11550_

2.3. Automating administration tasks

This chapter assumes that you have experience of administering WebSphere® MQ objects.

You might decide that it would be beneficial to your installation to automate some administration and monitoring tasks. You can automate administration tasks for both local and remote queue managers using programmable command format (PCF) commands.

This chapter describes:

- How to use programmable command formats to automate administration tasks in PCF commands.
- Support for Microsoft's Active Directory Service Interfaces (ADSI) in Active Directory Services Interfaces.

PCF commands

Parent topic: Administration using WebSphere MQ commands

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11560_

2.3.1. PCF commands

The purpose of WebSphere® MQ programmable command format (PCF) commands is to allow administration tasks to be programmed into an administration program. In this way, from a program you can manipulate queue manager objects (queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects), and even manipulate the queue managers themselves.

PCF commands cover the same range of functions provided by MQSC commands. You can write a program to issue PCF commands to any queue manager in the network from a single node. In this way, you can both centralize and automate administration tasks.

Each PCF command is a data structure that is embedded in the application data part of a WebSphere MQ message. Each command is sent to the target queue manager using the MQI function MQPUT in the same way as any other message. Providing the command server is running on the queue manager receiving the message, the command server interprets it as a command message and runs the command. To get the replies, the application issues an MQGET call and the reply data is returned in another data structure. The application can then process the reply and act accordingly.

Note: Unlike MQSC commands, PCF commands and their replies are not in a text format that you can read.

Briefly, these are some of the things needed to create a PCF command message:

Message descriptor

This is a standard WebSphere MQ message descriptor, in which:

- **Message type (*MsqType*) is MQMT_REQUEST.**
- **Message format (*Format*) is MQFMT_ADMIN.**

Application data

Contains the PCF message including the PCF header, in which:

- **The PCF message type (*Type*) specifies MQCFT_COMMAND.**
- **The command identifier specifies the command, for example, *Change Queue* (MQCMD_CHANGE_Q).**

For a complete description of the PCF data structures and how to implement them, see WebSphere MQ Programmable Command Formats and Administration Interface.

PCF object attributes

Escape PCFs

Using the MQAI to simplify the use of PCFs

Active Directory Services Interfaces

Parent topic: Automating administration tasks

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11570_

2.3.1.1. PCF object attributes

Object attributes in PCF are not limited to eight characters as they are for MQSC commands. They are shown in this book in italics. For example, the PCF equivalent of RQMNAME is *RemoteQMgrName*.

Parent topic: PCF commands

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11580_

2.3.1.2. Escape PCFs

Escape PCFs are PCF commands that contain MQSC commands within the message text. You can use PCFs to send

commands to a remote queue manager. For more information about using escape PCFs, see WebSphere MQ Programmable Command Formats and Administration Interface.

Parent topic: PCF commands

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11590_

2.3.1.3. Using the MQAI to simplify the use of PCFs

The MQAI is an administration interface to WebSphere® MQ that is available on the AIX®, HP-UX, Linux, Solaris, and Windows platforms.

It performs administration tasks on a queue manager through the use of *data bags*. Data bags allow you to handle properties (or parameters) of objects in a way that is easier than using PCFs.

Use the MQAI:

To simplify the use of PCF messages

The MQAI is an easy way to administer WebSphere MQ; you do not have to write your own PCF messages, avoiding the problems associated with complex data structures.

To pass parameters in programs written using MQI calls, the PCF message must contain the command and details of the string or integer data. To do this, you need several statements in your program for every structure, and memory space must be allocated. This task can be long and laborious.

Programs written using the MQAI pass parameters into the appropriate data bag and you need only one statement for each structure. The use of MQAI data bags removes the need for you to handle arrays and allocate storage, and provides some degree of isolation from the details of the PCF.

To implement self-administering applications and administration tools

For example, the Active Directory Services Interfaces provided by WebSphere MQ for Windows use the MQAI.

To handle error conditions more easily

It is difficult to get return codes back from PCF commands, but the MQAI makes it easier for the program to handle error conditions.

After you have created and populated your data bag, you can send an administration command message to the command server of a queue manager, using the mqExecute call, which waits for any response messages. The mqExecute call handles the exchange with the command server and returns responses in a *response bag*.

For more information about using the MQAI, and PCFs in general, see WebSphere MQ Programmable Command Formats and Administration Interface.

Parent topic: PCF commands

This build: January 26, 2011 11:54:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11600_

2.3.1.4. Active Directory Services Interfaces

Active Directory Service Interfaces (ADSI) support allows client applications to use a common set of Component Object Model (COM) interfaces to communicate with, and control, any application that implements them.

Unlike tools written using other WebSphere® MQ administration interfaces, those that use the ADSI are not limited to manipulating WebSphere MQ servers. The same tool can control Windows, Lotus Notes®, or any application implementing the ADSI.

WebSphere MQ support for the ADSI is implemented through the use of the IBMMQSeries namespace.

Any programming language that supports the COM interfaces can be used to implement ADSI clients.

For more information about the ADSI, visit the Microsoft website at http://www.microsoft.com.

For more information about Component Object Model (COM) interfaces, see WebSphere MQ Using the Component Object Model Interface.

Client connection channels in the Active Directory

Parent topic: PCF commands

This build: January 26, 2011 11:54:38

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11610_

2.3.1.4.1. Client connection channels in the Active Directory

On Windows systems that support the Active Directory, WebSphere® MQ publishes client connection channels in the Active Directory to provide dynamic client-server binding.

When client connection channel objects are defined, they are written into a client channel definition file, called AMQCLCHL.TAB by default. If the client connection channels use the TCP/IP protocol, the WebSphere MQ server also publishes them in the Active Directory. When the WebSphere MQ client determines how to connect to the server, it looks for a relevant client connection channel object definition using the following search order:

1. **MQCONNX MQCD data structure**
2. **MQSERVER environment variable**
3. **client channel definition file**
4. **Active Directory**

This order means that any current applications are not affected by any change. You can think of these entries in the Active Directory as records in the client channel definition file, and the WebSphere MQ client processes them in the same way. To configure and administer support for publishing client connection channel definitions in the Active Directory, use the setmqscp command, as described in setmqscp.
Parent topic: Active Directory Services Interfaces

This build: January 26, 2011 11:54:38

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11620_

2.4. Administering remote WebSphere MQ objects

This chapter tells you how to administer WebSphere® MQ objects on a remote queue manager using MQSC commands, and how to use remote queue objects to control the destination of messages and reply messages.

This chapter describes:

- Channels, clusters, and remote queuing
- Remote administration from a local queue manager
- Creating a local definition of a remote queue
- Using remote queue definitions as aliases
- Data conversion

Channels, clusters, and remote queuing

Remote administration from a local queue manager

Creating a local definition of a remote queue

Using remote queue definitions as aliases

Data conversion

Parent topic: Administration using WebSphere MQ commands

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11630_

2.4.1. Channels, clusters, and remote queuing

A queue manager communicates with another queue manager by sending a message and, if required, receiving back a response. The receiving queue manager could be:

- On the same machine
- On another machine in the same location (or even on the other side of the world)
- Running on the same platform as the local queue manager
- Running on another platform supported by WebSphere® MQ

These messages might originate from:

- User-written application programs that transfer data from one node to another
- User-written administration applications that use PCF commands, the MQAI, or the ADSI
- The WebSphere MQ Explorer.
- Queue managers sending:
  - Instrumentation event messages to another queue manager
  - MQSC commands issued from a runmqsc command in indirect mode (where the commands are run on another queue manager)

Before a message can be sent to a remote queue manager, the local queue manager needs a mechanism to detect the arrival of messages and transport them consisting of:

- At least one channel
- A transmission queue
- A channel initiator

For a remote queue manager to received a message, a listener is required.

A channel is a one-way communication link between two queue managers and can carry messages destined for any number of queues at the remote queue manager.

Each end of the channel has a separate definition. For example, if one end is a sender or a server, the other end must be a receiver or a requester. A simple channel consists of a *sender channel definition* at the local queue manager end and a *receiver channel definition* at the remote queue manager end. The two definitions must have the same name and together constitute a single message channel.

If you want the remote queue manager to respond to messages sent by the local queue manager, set up a second channel to send responses back to the local queue manager.

Use the MQSC command DEFINE CHANNEL to define channels. In this chapter, the examples relating to channels use the default channel attributes unless otherwise specified.

There is a message channel agent (MCA) at each end of a channel, controlling the sending and receiving of messages. The MCA takes messages from the transmission queue and puts them on the communication link between the queue managers.

A transmission queue is a specialized local queue that temporarily holds messages before the MCA picks them up and sends them to the remote queue manager. You specify the name of the transmission queue on a *remote queue definition*.

You can allow an MCA to transfer messages using multiple threads. This process is known as *pipelining*. Pipelining enables the MCA to transfer messages more efficiently, improving channel performance. See Attributes of channels for details of how to configure a channel to use pipelining.

Preparing channels and transmission queues for remote administration tells you how to use these definitions to set up remote administration.

For more information about setting up distributed queuing in general, see WebSphere MQ Intercommunication.

Remote administration using clusters

Parent topic: Administering remote WebSphere MQ objects

This build: January 26, 2011 11:54:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11640_

2.4.1.1. Remote administration using clusters

In a WebSphere® MQ network using distributed queuing, every queue manager is independent. If one queue manager needs to send messages to another queue manager, it must define a transmission queue, a channel to the remote queue manager, and a remote queue definition for every queue to which it wants to send messages.

A *cluster* is a group of queue managers set up in such a way that the queue managers can communicate directly with one another over a single network without complex transmission queue, channel, and queue definitions. Clusters can be set up easily, and typically contain queue managers that are logically related in some way and need to share data or applications. Even the smallest cluster reduces system administration costs.

Establishing a network of queue managers in a cluster involves fewer definitions than establishing a traditional distributed queuing environment. With fewer definitions to make, you can set up or change your network more quickly and easily, and reduce the risk of making an error in your definitions.

To set up a cluster, you need one cluster sender (CLUSSDR) and one cluster receiver (CLUSRCVR) definition for each queue manager. You do not need any transmission queue definitions or remote queue definitions. The principles of remote administration are the same when used within a cluster, but the definitions themselves are greatly simplified.

For more information about clusters, their attributes, and how to set them up, refer to WebSphere MQ Queue Manager Clusters.

Parent topic: Channels, clusters, and remote queuing

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11650_

2.4.2. Remote administration from a local queue manager

This section tells you how to administer a remote queue manager from a local queue manager using MQSC and PCF commands.

Preparing the queues and channels is essentially the same for both MQSC and PCF commands. In this book, the examples show MQSC commands, because they are easier to understand. For more information about writing administration programs using PCF commands, see WebSphere MQ Programmable Command Formats and Administration Interface.

You send MQSC commands to a remote queue manager either interactively or from a text file containing the commands. The remote queue manager might be on the same machine or, more typically, on a different machine. You can remotely administer queue managers in other WebSphere® MQ environments, including UNIX systems, Windows systems, i5/OS®, and z/OS®.

To implement remote administration, you must create specific objects. Unless you have specialized requirements, you should find that the default values (for example, for maximum message length) are sufficient.

Preparing queue managers for remote administration

Preparing channels and transmission queues for remote administration

Managing the command server for remote administration

Issuing MQSC commands on a remote queue manager

Recommendations for issuing commands remotely

If you have problems using MQSC commands remotely

Parent topic: Administering remote WebSphere MQ objects

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11660_

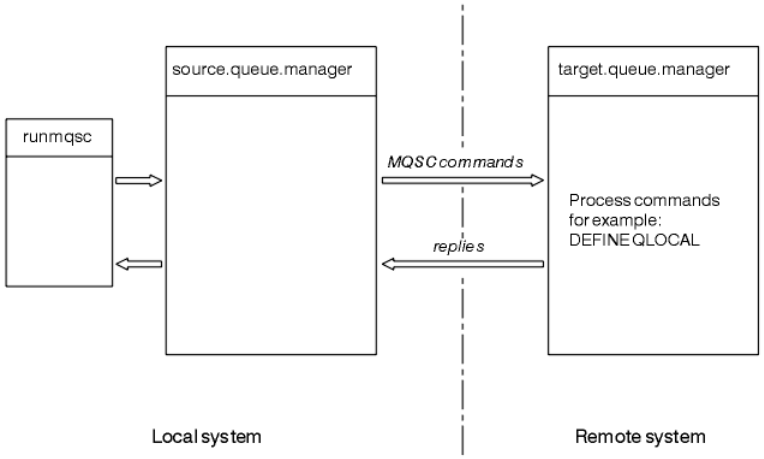
2.4.2.1. Preparing queue managers for remote administration

Figure 1 shows the configuration of queue managers and channels that you need for remote administration using

the runmqsc command. The object `source.queue.manager` is the source queue manager from which you can issue MQSC commands and to which the results of these commands (operator messages) are returned. The object `target.queue.manager` is the name of the target queue manager, which processes the commands and generates any operator messages.

Note: If you are using runmqsc with the -w option, `source.queue.manager` *must* be the default queue manager. For further information on creating a queue manager, see crtmqm.

Figure 1. Remote administration using MQSC commands



On both systems, if you have not already done so:

- Create the queue manager and the default objects, using the crtmqm command.
- Start the queue manager, using the strmqm command.

On the target queue manager:

- The command queue, SYSTEM.ADMIN.COMMAND.QUEUE, must be present. This queue is created by default when a queue manager is created.

You have to run these commands locally or over a network facility such as Telnet.

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:39

Notices | Trademarks | Downloads | Library | Support | Feedback

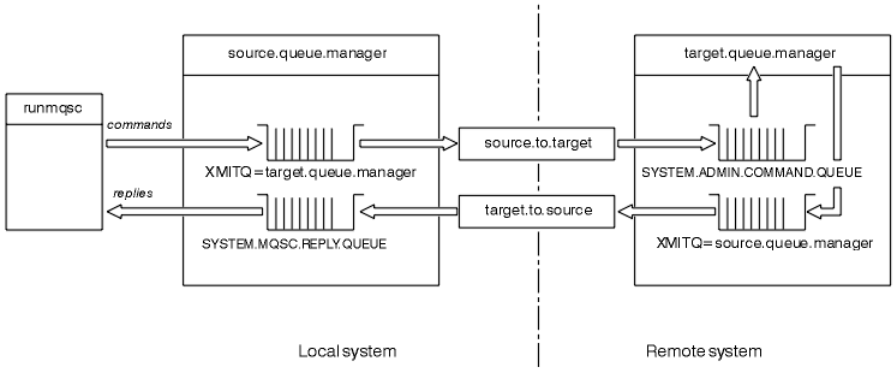
© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11670_

2.4.2.2. Preparing channels and transmission queues for remote administration

To run MQSC commands remotely, set up two channels, one for each direction, and their associated transmission queues. This example assumes that you are using TCP/IP as the transport type and that you know the TCP/IP address involved.

The channel `source.to.target` is for sending MQSC commands from the source queue manager to the target queue manager. Its sender is at `source.queue.manager` and its receiver is at `target.queue.manager`. The channel `target.to.source` is for returning the output from commands and any operator messages that are generated to the source queue manager. You must also define a transmission queue for each channel. This queue is a local queue that is given the name of the receiving queue manager. The XMITQ name must match the remote queue manager name in order for remote administration to work, unless you are using a queue manager alias. Figure 1 summarizes this configuration.

Figure 1. Setting up channels and queues for remote administration



See WebSphere MQ Intercommunication for more information about setting up channels.

- Defining channels, listeners, and transmission queues
- Starting the listeners and channels

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11680_

2.4.2.2.1. Defining channels, listeners, and transmission queues

On the source queue manager (`source.queue.manager`), issue the following MQSC commands to define the channels, listener, and the transmission queue:

1. **Define the sender channel at the source queue manager:**

```
DEFINE CHANNEL ('source.to.target') +
        CHLTYPE(SDR) +
        CONNAME (RHX5498) +
        XMITQ ('target.queue.manager') +
        TRPTYPE(TCP)
```

2. **Define the receiver channel at the source queue manager:**

```
DEFINE CHANNEL ('target.to.source') +
        CHLTYPE(RCVR) +
        TRPTYPE(TCP)
```

3. **Define the listener on the source queue manager:**

```
DEFINE LISTENER ('source.queue.manager') +
        TRPTYPE (TCP)
```

4. **Define the transmission queue on the source queue manager:**

```
DEFINE QLOCAL ('target.queue.manager') +
        USAGE (XMITQ)
```

Issue the following commands on the target queue manager (`target.queue.manager`), to create the channels, listener, and the transmission queue:

1. **Define the sender channel on the target queue manager:**

```
DEFINE CHANNEL ('target.to.source') +
        CHLTYPE(SDR) +
        CONNAME (RHX7721) +
        XMITQ ('source.queue.manager') +
        TRPTYPE(TCP)
```

2. **Define the receiver channel on the target queue manager:**

```
DEFINE CHANNEL ('source.to.target') +
        CHLTYPE(RCVR) +
        TRPTYPE(TCP)
```

3. Define the listener on the target queue manager:

```
DEFINE LISTENER ('target.queue.manager') +
        TRPTYPE (TCP)
```

4. Define the transmission queue on the target queue manager:

```
DEFINE QLOCAL ('source.queue.manager') +
        USAGE (XMITQ)
```

Note: The TCP/IP connection names specified for the CONNAME attribute in the sender channel definitions are for illustration only. This is the network name of the machine at the *other* end of the connection. Use the values appropriate for your network.

Parent topic: Preparing channels and transmission queues for remote administration

This build: January 26, 2011 11:54:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11690_

2.4.2.2.2. Starting the listeners and channels

Start both listeners by using the following MQSC commands:

1. **Start the listener on the source queue manager, `source.queue.manager`, by issuing the following MQSC command:**

```
START LISTENER ('source.queue.manager')
```

2. **Start the listener on the target queue manager, `target.queue.manager`, by issuing the following MQSC command:**

```
START LISTENER ('target.queue.manager')
```

Start both sender channels by using the following MQSC commands:

1. **Start the sender channel on the source queue manager, `source.queue.manager`, by issuing the following MQSC command:**

```
START CHANNEL ('source.to.target')
```

2. **Start the sender channel on the target queue manager, `target.queue.manager`, by issuing the following MQSC command:**

```
START CHANNEL ('target.to.source')
```

Automatic definition of channels

Parent topic: Preparing channels and transmission queues for remote administration

This build: January 26, 2011 11:54:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11700_

2.4.2.2.2.1. Automatic definition of channels

If WebSphere® MQ receives an inbound attach request and cannot find an appropriate receiver or server-connection channel, it creates a channel automatically. Automatic definitions are based on two default definitions supplied with WebSphere MQ: SYSTEM.AUTO.RECEIVER and SYSTEM.AUTO.SVRCONN.

You enable automatic definition of receiver and server-connection definitions by updating the queue manager object using the MQSC command, ALTER QMGR (or the PCF command Change Queue Manager).

For more information about creating channel definitions automatically, see WebSphere MQ Intercommunication. For information about automatically defining channels for clusters, see WebSphere MQ Queue Manager Clusters.

Parent topic: Starting the listeners and channels

This build: January 26, 2011 11:54:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11710_

2.4.2.3. Managing the command server for remote administration

Each queue manager can have a command server associated with it. A command server processes any incoming commands from remote queue managers, or PCF commands from applications. It presents the commands to the queue manager for processing and returns a completion code or operator message depending on the origin of the command.

A command server is mandatory for all administration involving PCF commands, the MQAI, and also for remote administration.

Note: For remote administration, ensure that the target queue manager is running. Otherwise, the messages containing commands cannot leave the queue manager from which they are issued. Instead, these messages are queued in the local transmission queue that serves the remote queue manager. Avoid this situation.

There are separate control commands for starting and stopping the command server. Providing the command server is running, users of WebSphere® MQ for Windows or WebSphere MQ for Linux (x86 platform) can perform the operations described in the following sections using the WebSphere MQ Explorer. For more information, see Administration using the WebSphere MQ Explorer.

Starting the command server

Displaying the status of the command server

Stopping a command server

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11720_

2.4.2.3.1. Starting the command server

Depending on the value of the queue manager attribute, *SCMDSERV*, the command server is either started automatically when the queue manager starts, or must be started manually. The value of the queue manager attribute can be altered using the MQSC command `ALTER QMGR` specifying the parameter `SCMDSERV`. By default, the command server is started automatically.

If *SCMDSERV* is set to MANUAL, start the command server using the command:

```
strmqcsv saturn.queue.manager
```

where `saturn.queue.manager` is the queue manager for which the command server is being started.

Parent topic: Managing the command server for remote administration

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11730_

2.4.2.3.2. Displaying the status of the command server

For remote administration, ensure that the command server on the target queue manager is running. If it is not running, remote commands cannot be processed. Any messages containing commands are queued in the target queue manager's command queue.

To display the status of the command server for a queue manager, issue the following MQSC command:

```
DISPLAY QMSTATUS CMDSERV
```

Parent topic: Managing the command server for remote administration

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11740_

2.4.2.3.3. Stopping a command server

To end the command server started by the previous example use the following command:

```
endmqcsv saturn.queue.manager
```

You can stop the command server in two ways:

- For a controlled stop, use the endmqcsv command with the -c flag, which is the default.
- For an immediate stop, use the endmqcsv command with the -i flag.

Note: Stopping a queue manager also ends the command server associated with it.

Parent topic: Managing the command server for remote administration

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11750_

2.4.2.4. Issuing MQSC commands on a remote queue manager

The command server *must* be running on the target queue manager, if it is going to process MQSC commands remotely. (This is not necessary on the source queue manager). For information on how to start the command server on a queue manager, see Starting the command server.

On the source queue manager, you can then run MQSC commands interactively in indirect mode by typing:

```
runmqsc -w 30 target.queue.manager
```

This form of the runmqsc command, with the -w flag, runs the MQSC commands in indirect mode, where commands are put (in a modified form) on the command server input queue and executed in order.

When you type in an MQSC command, it is redirected to the remote queue manager, in this case, `target.queue.manager`. The timeout is set to 30 seconds; if a reply is not received within 30 seconds, the following message is generated on the local (source) queue manager:

```
AMQ8416: MQSC timed out waiting for a response from the command server.
```

When you stop issuing MQSC commands, the local queue manager displays any timed-out responses that have arrived and discards any further responses.

The source queue manager defaults to the default local queue manager. If you specify the `-m` *LocalQmgrName* option in the runmqsc command, you can direct the commands to be issued by way of any local queue manager.

In indirect mode, you can also run an MQSC command file on a remote queue manager. For example:

```
runmqsc -w 60 target.queue.manager < mycomds.in > report.out
```

where `mycomds.in` is a file containing MQSC commands and `report.out` is the report file.

Working with queue managers on z/OS

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:39

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11760_

2.4.2.4.1. Working with queue managers on z/OS

You can issue MQSC commands to a z/OS® queue manager from a queue manager on the platforms described in this book. However, to do this, you must modify the runmqsc command and the channel definitions at the sender.

In particular, you add the -x flag to the runmqsc command on the source node to specify that the target queue manager is running under z/OS:

```
runmqsc -w 30 -x target.queue.manager
```

Parent topic: Issuing MQSC commands on a remote queue manager

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11770_

2.4.2.5. Recommendations for issuing commands remotely

When you are issuing commands on a remote queue manager:

1. Put the MQSC commands to be run on the remote system in a command file.
2. Verify your MQSC commands locally, by specifying the -v flag on the runmqsc command. You cannot use runmqsc to verify MQSC commands on another queue manager.
3. Check that the command file runs locally without error.
4. Run the command file against the remote system.

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11780_

2.4.2.6. If you have problems using MQSC commands remotely

If you have difficulty in running MQSC commands remotely, make sure that you have:

- Started the command server on the target queue manager.
- Defined a valid transmission queue.
- Defined the two ends of the message channels for both:
  - The channel along which the commands are being sent.
  - The channel along which the replies are to be returned.
- Specified the correct connection name (CONNAME) in the channel definition.
- Started the listeners before you started the message channels.
- Checked that the disconnect interval has not expired, for example, if a channel started but then shut down after some time. This is especially important if you start the channels manually.
- Sent requests from a source queue manager that do not make sense to the target queue manager (for example, requests that include parameters that are not supported on the remote queue manager).

See also Resolving problems with MQSC commands.

Parent topic: Remote administration from a local queue manager

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11790_

2.4.3. Creating a local definition of a remote queue

A local definition of a remote queue is a definition on a local queue manager that refers to a queue on a remote queue manager.

You do not have to define a remote queue from a local position, but the advantage of doing so is that applications can refer to the remote queue by its locally-defined name instead of having to specify a name that is qualified by the ID of the queue manager on which the remote queue is located.

Understanding how local definitions of remote queues work

An alternative way of putting messages on a remote queue

Using other commands with remote queues

Defining a transmission queue

Parent topic: Administering remote WebSphere MQ objects

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11800_

2.4.3.1. Understanding how local definitions of remote queues work

An application connects to a local queue manager and then issues an MQOPEN call. In the open call, the queue name specified is that of a remote queue definition on the local queue manager. The remote queue definition supplies the names of the target queue, the target queue manager, and optionally, a transmission queue. To put a message on the remote queue, the application issues an MQPUT call, specifying the handle returned from the MQOPEN call. The queue manager uses the remote queue name and the remote queue manager name in a transmission header at the start of the message. This information is used to route the message to its correct destination in the network.

As administrator, you can control the destination of the message by altering the remote queue definition.

Example

Parent topic: Creating a local definition of a remote queue

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11810_

2.4.3.1.1. Example

Purpose

How it works

Parent topic: Understanding how local definitions of remote queues work

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11820_

2.4.3.1.1.1. Purpose

An application needs to put a message on a queue owned by a remote queue manager.

Parent topic: Example

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11830_

2.4.3.1.1.2. How it works

The application connects to a queue manager, for example, `saturn.queue.manager`. The target queue is owned by another queue manager.

On the MQOPEN call, the application specifies these fields:

Field value	Description
ObjectName CYAN.REMOTE.QUEUE	Specifies the local name of the remote queue object. This defines the target queue and the target queue manager.
ObjectType (Queue)	Identifies this object as a queue.
ObjectQmgrName Blank or `saturn.queue.manager`	This field is optional. If blank, the name of the local queue manager is assumed. (This is the queue manager on which the remote queue definition exists.)

After this, the application issues an MQPUT call to put a message onto this queue.

On the local queue manager, you can create a local definition of a remote queue using the following MQSC commands:

```
DEFINE QREMOTE (CYAN.REMOTE.QUEUE) +
       DESCR ('Queue for auto insurance requests from the branches') +
       RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE) +
       RQMNAME (jupiter.queue.manager) +
       XMITQ (INQUOTE.XMIT.QUEUE)
```

where:

`QREMOTE (CYAN.REMOTE.QUEUE)`

Specifies the local name of the remote queue object. This is the name that applications connected to this queue manager must specify in the MQOPEN call to open the queue AUTOMOBILE.INSURANCE.QUOTE.QUEUE on the remote queue manager `jupiter.queue.manager`.

`DESCR ('Queue for auto insurance requests from the branches')`

Provides additional text that describes the use of the queue.

`RNAME (AUTOMOBILE.INSURANCE.QUOTE.QUEUE)`

Specifies the name of the target queue on the remote queue manager. This is the real target queue for messages sent by applications that specify the queue name CYAN.REMOTE.QUEUE. The queue AUTOMOBILE.INSURANCE.QUOTE.QUEUE must be defined as a local queue on the remote queue manager.

`RQMNAME (jupiter.queue.manager)`

Specifies the name of the remote queue manager that owns the target queue AUTOMOBILE.INSURANCE.QUOTE.QUEUE.

`XMITQ (INQUOTE.XMIT.QUEUE)`

Specifies the name of the transmission queue. This is optional; if the name of a transmission queue is not specified, a queue with the same name as the remote queue manager is used.

In either case, the appropriate transmission queue must be defined as a local queue with a *`Usage`* attribute specifying that it is a transmission queue (USAGE(XMITQ) in MQSC commands).

Parent topic: Example

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11840_

2.4.3.2. An alternative way of putting messages on a remote queue

Using a local definition of a remote queue is not the only way of putting messages on a remote queue. Applications can specify the full queue name, including the remote queue manager name, as part of the MQOPEN call. In this case, you do not need a local definition of a remote queue. However, this means that applications must either know, or have access to, the name of the remote queue manager at run time.

Parent topic: Creating a local definition of a remote queue

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa11850_

2.4.3.3. Using other commands with remote queues

You can use MQSC commands to display or alter the attributes of a remote queue object, or you can delete the remote queue object. For example:

- To display the remote queue's attributes:

```
DISPLAY QUEUE (CYAN.REMOTE.QUEUE)
```

- To change the remote queue to enable puts. This does not affect the target queue, only applications that specify this remote queue:

```
ALTER QREMOTE (CYAN.REMOTE.QUEUE) PUT(ENABLED)
```

- To delete this remote queue. This does not affect the target queue, only its local definition:

```
DELETE QREMOTE (CYAN.REMOTE.QUEUE)
```

Note: When you delete a remote queue, you delete only the local representation of the remote queue. You do not delete the remote queue itself or any messages on it.

Parent topic: Creating a local definition of a remote queue

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11860_

2.4.3.4. Defining a transmission queue

A transmission queue is a local queue that is used when a queue manager forwards messages to a remote queue manager through a message channel.

The channel provides a one-way link to the remote queue manager. Messages are queued at the transmission queue until the channel can accept them. When you define a channel, you must specify a transmission queue name at the sending end of the message channel.

The MQSC command attribute USAGE defines whether a queue is a transmission queue or a normal queue.

Default transmission queues

Parent topic: Creating a local definition of a remote queue

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11870_

2.4.3.4.1. Default transmission queues

When a queue manager sends messages to a remote queue manager, it identifies the transmission queue using the following sequence:

1. The transmission queue named on the XMITQ attribute of the local definition of a remote queue.
2. A transmission queue with the same name as the target queue manager. (This value is the default value on XMITQ of the local definition of a remote queue.)
3. The transmission queue named on the DEFXMITQ attribute of the local queue manager.

For example, the following MQSC command creates a default transmission queue on `source.queue.manager` for messages going to `target.queue.manager`:

```
DEFINE QLOCAL ('target.queue.manager') +
       DESCR ('Default transmission queue for target qm') +
       USAGE (XMITQ)
```

Applications can put messages directly on a transmission queue, or indirectly through a remote queue definition. See also Creating a local definition of a remote queue.

Parent topic: Defining a transmission queue

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11880_

2.4.4. Using remote queue definitions as aliases

In addition to locating a queue on another queue manager, you can also use a local definition of a remote queue for both:

- Queue manager aliases
- Reply-to queue aliases

Both types of alias are resolved through the local definition of a remote queue.

You must set up the appropriate channels for the message to arrive at its destination.

Queue manager aliases

Reply-to queue aliases

Parent topic: Administering remote WebSphere MQ objects

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11890_

2.4.4.1. Queue manager aliases

An alias is the process by which the name of the target queue manager, as specified in a message, is modified by a queue manager on the message route. Queue manager aliases are important because you can use them to control the destination of messages within a network of queue managers.

You do this by altering the remote queue definition on the queue manager at the point of control. The sending application is not aware that the queue manager name specified is an alias.

For more information about queue manager aliases, see WebSphere MQ Intercommunication.

Parent topic: Using remote queue definitions as aliases

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11900_

2.4.4.2. Reply-to queue aliases

Optionally, an application can specify the name of a reply-to queue when it puts a *request message* on a queue.

If the application that processes the message extracts the name of the reply-to queue, it knows where to send the *reply message*, if required.

A reply-to queue alias is the process by which a reply-to queue, as specified in a request message, is altered by a queue manager on the message route. The sending application is not aware that the reply-to queue name specified is an alias.

A reply-to queue alias lets you alter the name of the reply-to queue and optionally its queue manager. This in turn lets you control which route is used for reply messages.

For more information about request messages, reply messages, and reply-to queues, see the WebSphere MQ Application Programming Guide.

For more information about reply-to queue aliases, see WebSphere MQ Intercommunication.

Parent topic: Using remote queue definitions as aliases

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11910_

2.4.5. Data conversion

Message data in WebSphere® MQ defined formats (also known as *built-in formats*) can be converted by the queue manager from one coded character set to another, provided that both character sets relate to a single language or a group of similar languages.

For example, conversion between coded character sets with identifiers (CCSIDs) 850 and 500 is supported, because both apply to Western European languages.

For EBCDIC newline (NL) character conversions to ASCII, see All queue managers.

Supported conversions are defined in the WebSphere MQ Application Programming Reference.

When a queue manager cannot convert messages in built-in formats

File ccsid.tbl

Converting messages in user-defined formats

Changing the queue manager CCSID

Parent topic: Administering remote WebSphere MQ objects

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11920_

2.4.5.1. When a queue manager cannot convert messages in built-in formats

The queue manager cannot automatically convert messages in built-in formats if their CCSIDs represent different national-language groups. For example, conversion between CCSID 850 and CCSID 1025 (which is an EBCDIC coded character set for languages using Cyrillic script) is not supported because many of the characters in one coded character set cannot be represented in the other. If you have a network of queue managers working in different national languages, and data conversion among some of the coded character sets is not supported, you can enable a default conversion. Default data conversion is described in Default data conversion.

Parent topic: Data conversion

This build: January 26, 2011 11:54:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11930_

2.4.5.2. File ccsid.tbl

The file ccsid.tbl is used for the following purposes:

- **In WebSphere® MQ for Windows it records all the supported code sets. In UNIX systems the supported code sets are held internally by the operating system.**
- **It specifies any additional code sets. To specify additional code sets, you need to edit ccsid.tbl (guidance on how to do this is provided in the file).**
- **It specifies any default data conversion.**

You can update the information recorded in ccsid.tbl; you might want to do this if, for example, a future release of your operating system supports additional coded character sets.

In WebSphere MQ for Windows, ccsid.tbl is located in directory `C:\Program Files\IBM\WebSphere MQ\conv\table` by default.

In WebSphere MQ for UNIX systems, ccsid.tbl is located in directory `/var/mqm/conv/table`.

Default data conversion

Parent topic: Data conversion

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11940_

2.4.5.2.1. Default data conversion

If you set up channels between two machines on which data conversion is not normally supported, you must enable default data conversion for the channels to work.

To enable default data conversion, edit the ccsid.tbl file to specify a default EBCDIC CCSID and a default ASCII CCSID. Instructions on how to do this are included in the file. You must do this on all machines that will be connected using the channels. Restart the queue manager for the change to take effect.

The default data-conversion process is as follows:

- If conversion between the source and target CCSIDs is not supported, but the CCSIDs of the source and target environments are either both EBCDIC or both ASCII, the character data is passed to the target application without conversion.
- If one CCSID represents an ASCII coded character set, and the other represents an EBCDIC coded character set, WebSphere® MQ converts the data using the default data-conversion CCSIDs defined in ccsid.tbl.

Note: Try to restrict the characters being converted to those that have the same code values in the coded character set specified for the message and in the default coded character set. If you use only the set of characters that is valid for WebSphere MQ object names (as defined in Names of WebSphere MQ objects) you will, in general, satisfy this requirement. Exceptions occur with EBCDIC CCSIDs 290, 930, 1279, and 5026 used in Japan, where the lowercase characters have different codes from those used in other EBCDIC CCSIDs.

Parent topic: File ccsid.tbl

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11950_

2.4.5.3. Converting messages in user-defined formats

The queue manager cannot convert messages in user-defined formats from one coded character set to another. If you need to convert data in a user-defined format, you must supply a data-conversion exit for each such format. Do not use default CCSIDs to convert character data in user-defined formats. For more information about converting data in user-defined formats and about writing data conversion exits, see the WebSphere MQ Application Programming Guide.

Parent topic: Data conversion

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11960_

2.4.5.4. Changing the queue manager CCSID

When you have used the CCSID attribute of the ALTER QMGR command to change the CCSID of the queue manager, stop and restart the queue manager to ensure that all running applications, including the command server and channel programs, are stopped and restarted.

This is necessary, because any applications that are running when the queue manager CCSID is changed continue to use the existing CCSID.

Parent topic: Data conversion

This build: January 26, 2011 11:54:40

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11970_

3. Administration using the WebSphere MQ Explorer

The WebSphere® MQ Explorer allows you to perform local or remote administration of your network from a computer running Windows, or Linux (x86 platform)

WebSphere MQ for Windows, and WebSphere MQ for Linux (x86 platform) provide an administration interface called the WebSphere MQ Explorer to perform administration tasks as an alternative to using control or MQSC commands. Comparing command sets shows you what you can do using the WebSphere MQ Explorer.

WebSphere MQ Explorer runs on all supported Windows platforms, and Linux (x86 platform only). If used in a 64-bit Windows environment, it runs only in 32-bit mode.

The WebSphere MQ Explorer allows you to perform local or remote administration of your network from a computer running Windows, or Linux (x86 platform), by pointing the WebSphere MQ Explorer at the queue managers and clusters you are interested in. The platforms and levels of WebSphere MQ that can be administered using the WebSphere MQ Explorer are described in Remote queue managers.

To configure remote WebSphere MQ queue managers so that WebSphere MQ Explorer can administer them, see Required definitions for administration.

It allows you to perform tasks, typically associated with setting up and fine tuning the working environment for WebSphere MQ, either locally or remotely within a Windows or Linux (x86 platform) system domain.

On Linux, the WebSphere MQ Explorer might fail to start if you have more than one Eclipse installation. If this happens, start the WebSphere MQ Explorer using a different user ID to the one you use for the other Eclipse installation.

»On Linux, to start the WebSphere MQ Explorer successfully, you must be able to write a file to your home directory, and the home directory must exist.«

This chapter describes:

- **What you can do with the WebSphere MQ Explorer**
- **Setting up the WebSphere MQ Explorer**
- **Using the WebSphere MQ Explorer**
- **Security on Windows**

What you can do with the WebSphere MQ Explorer

Setting up the WebSphere MQ Explorer

Using the WebSphere MQ Explorer

Security on Windows

Extending the WebSphere MQ Explorer

Parent topic: System Administration Guide

This build: January 26, 2011 11:54:41

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa11990_

3.1. What you can do with the WebSphere MQ Explorer

With the WebSphere® MQ Explorer, you can:

- **Create and delete a queue manager (on your local machine only).**
- **Start and stop a queue manager (on your local machine only).**
- **Define, display, and alter the definitions of WebSphere MQ objects such as queues and channels.**
- **Browse the messages on a queue.**

- Start and stop a channel.
- View status information about a channel, listener, queue, or service objects.
- View queue managers in a cluster.
- Check to see which applications, users, or channels have a particular queue open.
- Create a new queue manager cluster using the *Create New Cluster* wizard.
- Add a queue manager to a cluster using the *Add Queue Manager to Cluster* wizard.
- Manage the authentication information object, used with Secure Sockets Layer (SSL) channel security.
- Create and delete channel initiators, trigger monitors, and listeners.
- Start or stop the command servers, channel initiators, trigger monitors, and listeners.
- Set specific services to start up automatically when a queue manager is started.
- Modify the properties of queue managers.
- Change the local default queue manager.
- Invoke the ikeyman GUI to manage secure sockets layer (SSL) certificates, associate certificates with queue managers, and configure and setup certificate stores (on your local machine only).
- Create JMS objects from WebSphere MQ objects, and WebSphere MQ objects from JMS objects.
- Create a JMS Connection Factory for any of the currently supported types.
- Modify the parameters for any service, such as the TCP port number for a listener, or a channel initiator queue name.
- Start or stop the service trace.

You perform administration tasks using a series of *Content Views* and *Property dialogs*.

Content View

A Content View is a panel that can display the following:

- Attributes, and administrative options relating to WebSphere MQ itself.
- Attributes, and administrative options relating to one or more related objects.
- Attributes, and administrative options for a cluster.

Property dialogs

A property dialog is a panel that displays attributes relating to an object in a series of fields, some of which you can edit.

You navigate through the WebSphere MQ Explorer using the *Navigator view*. The Navigator allows you to select the Content View you require.

Remote queue managers

Deciding whether to use the WebSphere MQ Explorer

Parent topic: Administration using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12000_

3.1.1. Remote queue managers

From a Windows or Linux (x86 platform) system, the WebSphere® MQ Explorer can connect to all supported queue managers with the following exceptions:

- WebSphere MQ for z/OS® queue managers earlier than Version 6.0.
- Currently supported MQSeries® V2 queue managers.

The WebSphere MQ Explorer handles the differences in the capabilities between the different command levels and platforms. However, if it encounters an attribute that it does not recognize, the attribute will not be visible.

If you intend to remotely administer a V6.0 or later queue manager on Windows using the WebSphere MQ Explorer on a WebSphere MQ V5.3 computer, you must install Fix Pack 9 (CSD9) or later on your WebSphere MQ for Windows V5.3 computer.

If you intend to remotely administer a V5.3 queue manager on iSeries® using the WebSphere MQ Explorer on a WebSphere MQ V6.0 or later computer, you must install Fix Pack 11 (CSD11) or later on your WebSphere MQ for iSeries V5.3 computer. This fix pack corrects connection problems between the WebSphere MQ Explorer and the

iSeries queue manager.

Parent topic: What you can do with the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12010_

3.1.2. Deciding whether to use the WebSphere MQ Explorer

When deciding whether to use the WebSphere® MQ Explorer at your installation, consider the following points:

Object names

If you use lowercase names for queue managers and other objects with the WebSphere MQ Explorer, when you work with the objects using MQSC commands, you must enclose the object names in single quotes, or WebSphere MQ will not recognize them.

Large queue managers

The WebSphere MQ Explorer works best with small queue managers. If you have a large number of objects on a single queue manager, you might experience delays while the WebSphere MQ Explorer extracts the required information to present in a view.

Clusters

WebSphere MQ clusters can potentially contain hundreds or thousands of queue managers. The WebSphere MQ Explorer presents the queue managers in a cluster using a tree structure. The physical size of a cluster does not affect the speed of the WebSphere MQ Explorer dramatically because the explorer does not connect to the queue managers in the cluster until you select them.

Parent topic: What you can do with the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12020_

3.2. Setting up the WebSphere MQ Explorer

This section outlines the steps you need to take to set up the WebSphere® MQ Explorer.

Prerequisite software

Required definitions for administration

Cluster membership

Security

Data conversion

Parent topic: Administration using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12030_

3.2.1. Prerequisite software

Before you can use the WebSphere® MQ Explorer, you must have the following installed on your computer:

- The Eclipse platform (installed as part of WebSphere MQ for Windows or WebSphere MQ for Linux (x86 platform))

The WebSphere MQ Explorer can connect to remote queue managers using the TCP/IP communication protocol

only.

Parent topic: Setting up the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12040_

3.2.2. Required definitions for administration

Ensure that you have satisfied the following requirements before trying to use the WebSphere® MQ Explorer. Check that:

1. A command server is running on every remotely administered queue manager.
2. A suitable TCP/IP listener object must be running on every remote queue manager. This can be the WebSphere MQ listener or, on UNIX systems, the inetd daemon.
3. A server-connection channel, by default named SYSTEM.ADMIN.SVRCONN, exists on all remote queue managers.
   You can create the channel using the following MQSC command:

   ```
   DEFINE CHANNEL(SYSTEM.ADMIN.SVRCONN) CHLTYPE(SVRCONN)
   ```

   This command creates a basic channel definition. If you want a more sophisticated definition (to set up security, for example), you need additional parameters.
   »The Client Attachment feature (CAF) is an option of WebSphere MQ for z/OS that supports the attachment of clients to z/OS«. If you require more than five attaches of the WebSphere MQ Explorer you need to install the Client Attachment feature (CAF). With the CAF installed, you can set the MAXINST attribute on the SYSTEM.ADMIN.SVRCONN channel from zero through 999 999 999.
   If you do not have the Client Attachment feature (CAF) installed, the MAXINST attribute can be set from zero to five only on the SYSTEM.ADMIN.SVRCONN channel. A value greater than five is interpreted as zero without the CAF installed.
4. The system queue, SYSTEM.MQEXPLORER.REPLY.MODEL, must exist.

Parent topic: Setting up the WebSphere MQ Explorer

This build: January 26, 2011 11:54:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12050_

3.2.3. Cluster membership

If a queue manager is a member of a cluster, then the cluster tree node will be populated automatically.

If queue managers become members of clusters while the WebSphere® MQ Explorer is running, then you must maintain the WebSphere MQ Explorer with up-to-date administration data about clusters so that it can communicate effectively with them and display correct cluster information when requested. In order to do this, the WebSphere MQ Explorer needs the following information:

- The name of a repository queue manager
- The connection name of the repository queue manager if it is on a remote queue manager

With this information, the WebSphere MQ Explorer can:

- Use the repository queue manager to obtain a list of queue managers in the cluster.
- Administer the queue managers that are members of the cluster and are on supported platforms and command levels.

Administration is not possible if:

- The chosen repository becomes unavailable. The WebSphere MQ Explorer does not automatically switch to an alternative repository.
- The chosen repository cannot be contacted over TCP/IP.
- The chosen repository is running on a queue manager that is running on a platform and command level not supported by the WebSphere MQ Explorer.

The cluster members that can be administered can be local, or they can be remote if they can be contacted using TCP/IP. The WebSphere MQ Explorer connects to local queue managers that are members of a cluster directly, without using a client connection.

Parent topic: Setting up the WebSphere MQ Explorer

This build: January 26, 2011 11:54:42

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12060_

3.2.4. Security

If you are using WebSphere® MQ in an environment where it is important for you to control user access to particular objects, you might need to consider the security aspects of using the WebSphere MQ Explorer.

Authorization to use the WebSphere MQ Explorer
Any user can use the WebSphere MQ Explorer, but certain authorities are required to connect, access, and manage queue managers.

Security for connecting to remote queue managers

Using a security exit with the WebSphere MQ Explorer
You can specify a default security exit and queue manager specific security exits using the WebSphere MQ Explorer.

Using SSL security

Using the WebSphere MQ Explorer to connect to a remote queue manager using SSL enabled MQI channels

Connecting through another queue manager

Parent topic: Setting up the WebSphere MQ Explorer

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12070_

3.2.4.1. Authorization to use the WebSphere MQ Explorer

Any user can use the WebSphere® MQ Explorer, but certain authorities are required to connect, access, and manage queue managers.

To perform local administrative tasks using the WebSphere MQ Explorer, a user is required to have the necessary authority to perform the administrative tasks. If the user is a member of the mqm group, the user has authority to perform all local administrative tasks.

To connect to a remote queue manager and perform remote administrative tasks using the WebSphere MQ Explorer, the user executing the WebSphere MQ Explorer is required to have the following authorities:

- **CONNECT authority on the target queue manager object**
- **INQUIRE authority on the target queue manager object**
- **DISPLAY authority to the target queue manager object**
- **INQUIRE authority to the queue, SYSTEM.MQEXPLORER.REPLY.MODEL**
- **DISPLAY authority to the queue, SYSTEM.MQEXPLORER.REPLY.MODEL**
- **INPUT authority to the queue, SYSTEM.MQEXPLORER.REPLY.MODEL**
- **OUTPUT authority to the queue, SYSTEM.ADMIN.COMMAND.QUEUE**
- **Authority to perform the action selected**

Note: »INPUT authority relates to input to the user from a queue (a get operation). OUTPUT authority relates to output from the user to a queue (a put operation). «

To connect to a remote queue manager on WebSphere MQ for z/OS® and perform remote administrative tasks using the WebSphere MQ Explorer, the following must be provided:

- **A RACF® profile for the system queue, SYSTEM.MQEXPLORER.REPLY.MODEL**
- **A RACF profile for the queues, AMQ.MQEXPLORER.***

In addition, the user executing the WebSphere MQ Explorer is required to have the following authorities:

- RACF UPDATE authority to the system queue, SYSTEM.MQEXPLORER.REPLY.MODEL
- RACF UPDATE authority to the queues, AMQ.MQEXPLORER.*
- CONNECT authority on the target queue manager object
- Authority to perform the action selected
- READ authority to all the hlq.DISPLAY.object profiles in the MQCMDS class

For information on how to grant authority to WebSphere MQ objects, see Authority to work with WebSphere MQ objects on UNIX systems and Windows.

If a user attempts to perform an operation that they are not authorized to perform, the target queue manager invokes authorization failure procedures and the operation fails.

The default filter in the WebSphere MQ Explorer is to display all WebSphere MQ objects. If there are any WebSphere MQ objects that a user does not have DISPLAY authority to, authorization failures are generated. If authority events are being recorded, restrict the range of objects that are displayed to those objects that the user has DISPLAY authority to.

Parent topic: Security

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12080_

3.2.4.2. Security for connecting to remote queue managers

The WebSphere® MQ Explorer connects to remote queue managers as an MQI client application. This means that each remote queue manager must have a definition of a server-connection channel and a suitable TCP/IP listener. If you do not specify a non-blank value for the MCAUSER attribute of the channel, or use a security exit, it is possible for a malicious application to connect to the same server connection channel and gain access to the queue manager objects with unlimited authority.

The default value of the MCAUSER attribute is the local userId. If you specify a non-blank user name as the MCAUSER attribute of the server connection channel, all programs connecting to the queue manager using this channel run with the identity of the named user and have the same level of authority.

Parent topic: Security

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12090_

3.2.4.3. Using a security exit with the WebSphere MQ Explorer

You can specify a default security exit and queue manager specific security exits using the WebSphere® MQ Explorer.

You can define a default security exit, which can be used for all new client connections from the WebSphere MQ Explorer. This default exit can be overridden at the time a connection is made. You can also define a security exit for a single queue manager or a set of queue managers, which takes effect when a connection is made. You specify exits using the WebSphere MQ Explorer. For more information, see the WebSphere MQ Help Center.

Parent topic: Security

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12100_

3.2.4.4. Using SSL security

The WebSphere® MQ Explorer connects to remote queue managers using an MQI channel. If you want to secure the MQI channel using SSL security, you must establish the channel using a client channel definition table. For information how to establish an MQI channel using a client channel definition table, see the WebSphere MQ Clients book.

Parent topic: Security

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12110_

3.2.4.5. Using the WebSphere MQ Explorer to connect to a remote queue manager using SSL enabled MQI channels

On the system hosting the remote queue manager:

1. Define a server connection and client connection pair of channels, and specify the appropriate value for the *SSLCIPH* variable on the server connection on both channels. For more information on the *SSLCIPH* variable, see Protecting channels with SSL
2. Send the channel definition table `AMQCLCHL.TAB`, which is found in the queue manager's `@ipcc` directory, to the system hosting the WebSphere MQ Explorer. To do this you can use the File Transfer Application in binary mode.
3. Start a TCP/IP listener on a designated port.
4. Place both the CA and personal SSL certificates into the queue manager's SSL directory:
   - `/var/mqm/qmgrs/+QMNAME+/SSL` for UNIX systems
   - `C:\Program Files\WebSphere MQ\qmgrs\+QMNAME+\SSL` for Windows systems
     Where `+QMNAME+` is a token representing the name of the queue manager.
5. Create a key database file of type CMS named `key.kdb`, and stash the password in a file either by checking the option in the iKeyman GUI, or by using the `-stash` option with the gsk7cmd, runmqckm ,or gsk7capicmd commands.
6. Add the CA certificates to the key database created in the previous step.
7. Import the personal certificate for the queue manager into the key database.

For more detailed information on working with the Secure Sockets Layer on Windows systems, see the WebSphere MQ Security book.

On the system hosting the WebSphere MQ Explorer:

1. Create a key database file of type JKS named `key.jks`. Set a password for this key database file.
   The WebSphere MQ Explorer uses Java key store files (JKS) for SSL security, and so the key store file being created for configuring SSL for the WebSphere MQ Explorer must match this.
2. Add the CA certificates to the key database created in the previous step.
3. Import the personal certificate for the queue manager into the key database.
4. Start the WebSphere MQ Explorer either by using the start menu in Windows, or by running the strmqcfg command.
5. From the WebSphere MQ Explorer toolbar, click Window -> Preferences, then expand WebSphere MQ Explorer and click SSL Client Certificate Stores. Enter the name of, and password for, the JKS file created in step 1 in both the Trusted Certificate Store and the Personal Certificate Store, then click OK.
6. Close the Preferences window, and right-click Queue Managers. Click Show/Hide Queue Managers, and then click Add on the Show/Hide Queue Managers screen.
7. Type the name of the queue manager, and select the Connect directly option. Click next.
8. Select Use client channel definition table and specify the location of the channel table file that you transferred from the remote queue manager in step 2 on the system hosting the remote queue manager.
9. Click Finish. You can now access the remote queue manager from the WebSphere MQ Explorer.

Parent topic: Security

This build: January 26, 2011 11:54:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12120_

3.2.4.6. Connecting through another queue manager

The WebSphere® MQ Explorer allows you to connect to a queue manager through an intermediate queue manager, to which the WebSphere MQ Explorer is already connected. In this case, the WebSphere MQ Explorer puts PCF command messages to the intermediate queue manager, specifying the following:

- The *ObjectQMgrName* parameter in the object descriptor (MQOD) as the name of the target queue manager. For

more information on queue name resolution, see the WebSphere MQ Application Programming Guide.

- The *UserIdentifier* parameter in the message descriptor (MQMD) as the local userId.

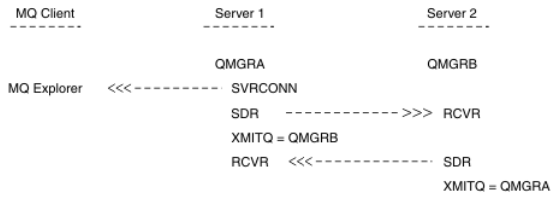
If the connection is then used to connect to the target queue manager via an intermediate queue manager, the userId is flowed in the *UserIdentifier* parameter of the message descriptor (MQMD) again. In order for the MCA listener on the target queue manager to accept this message, either the MCAUSER attribute must be set, or the userId must already exist with put authority.

The command server on the target queue manager puts messages to the transmission queue specifying the userId in the *UserIdentifier* parameter in the message descriptor (MQMD). For this put to succeed the userId must already exist on the target queue manager with put authority.

The following example shows you how to connect a queue manager, through an intermediate queue manager, to the WebSphere MQ Explorer.

Establish a remote administration connection to a queue manager. Verify that the:

- Queue manager on the server is active and has a server-connection channel (SVRCONN) defined.
- Listener is active.
- Command server is active.
- SYSTEM.MQ EXPLORER.REPLY.MODEL queue has been created and that you have sufficient authority.
- Queue manager listeners, command servers, and sender channels are started.



In this example:

- WebSphere MQ Explorer is connected to queue manager QMGRA (running on Server1) using a client connection.
- Queue manager QMGRB on Server2 can be now connected to WebSphere MQ Explorer through an intermediate queue manager (QMGRA)
- When connecting to QMGRB with WebSphere MQ Explorer, select QMGRA as the intermediate queue manager

In this situation, there is no direct connection to QMGRB from WebSphere MQ Explorer; the connection to QMGRB is through QMGRA.

Queue manager QMGRB on Server2 is connected to QMGRA on Server1 using sender-receiver channels. The channel between QMGRA and QMGRB must be set up in such a way that remote administration is possible; see Preparing channels and transmission queues for remote administration.

Parent topic: Security

This build: January 26, 2011 11:54:43

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12130_

3.2.5. Data conversion

The WebSphere® MQ Explorer works in CCSID 1208 (UTF-8). This enables the WebSphere MQ Explorer to display the data from remote queue managers correctly. Whether connecting to a queue manager directly, or by using an intermediate queue manager, the WebSphere MQ Explorer requires all incoming messages to be converted to CCSID 1208 (UTF-8).

An error message is issued if you try to establish a connection between the WebSphere MQ Explorer and a queue manager with a CCSID that the WebSphere MQ Explorer does not recognize.

Supported conversions are described in the WebSphere MQ Application Programming Reference manual.

Parent topic: Setting up the WebSphere MQ Explorer

This build: January 26, 2011 11:54:43

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa12140_

3.3. Using the WebSphere MQ Explorer

This section explains how to perform the following actions:

- Use the WebSphere® MQ Explorer to show or hide queue managers
- Starting the WebSphere MQ Explorer in standalone mode or Eclipse workbench mode

Showing and hiding queue managers and clusters

Starting the WebSphere MQ Explorer in standalone mode or Eclipse workbench mode

Parent topic: Administration using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12150_

3.3.1. Showing and hiding queue managers and clusters

The WebSphere® MQ Explorer can display more than one queue manager at a time. The *Show/Hide Queue Manager* panel (selectable from the context menu for the Queue Managers tree node) allows you to choose whether you display information on another (remote) machine. Local queue managers are detected automatically.

To show a remote queue manager:

1. Right-click the Queue Managers tree node, then select *Show/Hide Queue Managers....*
2. Click Add.... The Show/Hide Queue Managers panel is displayed.
3. Enter the name of the remote queue manager and the host name or IP address in the fields provided. The host name or IP address is used to establish a client connection to the remote queue manager using either its default server connection channel, SYSTEM.ADMIN.SVRCONN, or a user defined server connection channel.
4. Click Finish.

The Show/Hide Queue Managers panel also displays a list of all visible queue managers, and allows you to hide queue managers from the navigation view.

If the WebSphere MQ Explorer displays a queue manager that is a member of a cluster, the cluster is detected, and displayed automatically.

To export the list of remote queue managers from this panel:

1. Close the Show/Hide Queue Managers panel.
2. Right click the top IBM WebSphere MQ tree node in the Navigation pane of the WebSphere MQ Explorer, then select *Export MQ Explorer Settings....*
3. Select Remote queue manager connection information.
4. Select a file to store the exported settings in.
5. Finally, click OK to export the remote queue manager connection information to the specified file.

To import a list of remote queue managers:

1. Right click the top IBM WebSphere MQ tree node in the Navigation pane of the WebSphere MQ Explorer, then select *Import MQ Explorer Settings....*
2. Click Browse, and navigate to the path of the file that contains the remote queue manager connection information.
3. Click Open. If the file contains a list of remote queue managers, the Remote queue manager connection information box will be selected.
4. Finally, click OK to import the remote queue manager connection information into the WebSphere MQ Explorer.

Parent topic: Using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12160_

3.3.2. Starting the WebSphere MQ Explorer in standalone mode or Eclipse workbench mode

By default, the WebSphere® MQ Explorer starts up in standalone mode using the Eclipse Rich Client Platform (RCP). The tools you need to configure WebSphere MQ will be visible, and other Eclipse tools will be hidden.

If you want to use the other perspectives available in Eclipse, which include the Eclipse Workbench toolbar and menu options, you can set WebSphere MQ to start up in an Eclipse Workbench. To do this, click Window -> Preferences and then select WebSphere MQ Explorer from the list. In the Startup section, select in an Eclipse Workbench and click OK. The change will take effect the next time you start WebSphere MQ Explorer.

Parent topic: Using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12190_

3.4. Security on Windows

The WebSphere® MQ Explorer uses *Component Object Model* (COM) technology to communicate between servers and between processes on a server.

The COM server application, *AMQMSRVN*, is shared between any client processes that need to use the WebSphere MQ COM Services.

Because AMQMSRVN must be shared between non-interactive and interactive logon sessions, you must launch it under a special user account. This special user account is called MUSR_MQADMIN. When you install WebSphere MQ and run the Prepare WebSphere MQ Wizard for the first time, it creates a local user account for AMQMSRVN called MUSR_MQADMIN with the required settings and permissions. The password for MUSR_MQADMIN is randomly generated when the account is created, and used to configure the logon environment for AMQMSRVN. The generated password does not expire.

This WebSphere MQ account will not be affected by any account policies that are set up on the system to require that account passwords are changed after a certain period of time.

The password is not known outside this *onetime* processing and is stored by the Windows operating system in a secure part of the Registry.

Using Active directory (Windows only)

Controlling access (Windows only)

Changing the password of the AMQMSRVN user account

WebSphere MQ coordinating with DB2 as the resource manager

Parent topic: Administration using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12200_

3.4.1. Using Active directory (Windows only)

In some network configurations, where user accounts are defined on domain controllers that are using Active Directory, the local user account MUSR_MQADMIN might not have the authority it requires to query the group membership of other domain user accounts. The Prepare WebSphere® MQ Wizard identifies whether this is the case by carrying out tests and asking the user questions about the network configuration. If the local user account MUSR_MQADMIN does not have the required authority, the Prepare WebSphere MQ Wizard prompts the user for the account details of a domain user account with particular user rights. For the user rights that the domain user account requires see User rights required for AMQMSRVN. Once the user has entered valid account details for the domain user account into the Prepare WebSphere MQ Wizard, it configures AMQMSRVN to run under this account instead of the local user account MUSR_MQADMIN. The account details are held in the secure part of the Registry and cannot be read by users.

When the service is running, AMQMSRVN is launched and remains running for as long as the service is running. A

WebSphere MQ administrator who logs onto the server after AMQMSRVN is launched can use the WebSphere MQ Explorer to administer queue managers on the server. This connects the WebSphere MQ Explorer to the existing AMQMSRVN process. These two actions need different levels of permission before they can work:

- **The launch process requires a launch permission.**
- **The WebSphere MQ administrator requires Access permission.**

User rights required for AMQMSRVN

Changing the user name associated with WebSphere MQ Services

Parent topic: Security on Windows

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12210_

3.4.1.1. User rights required for AMQMSRVN

The following table details the user rights required for the domain user account under which WebSphere® MQ and specifically the AMQMSRVN DCOM object run.

Table 1. User rights required to launch AMQMSRVN

Logon as batch job	Enables WebSphere MQ Services COM server to run under this user account.
Logon as service	Enables users to set the WebSphere MQ service to logon using the configured account.
Shut down the system	Allows the WebSphere MQ Service to restart the server if configured to do so when recovery of a service fails.
Debug programs	Enables WebSphere MQ to contact processes that are secured, such as ASP and IIS applications.
Increase quotas	Required for operating system **CreateProcessAsUser** call.
Act as part of the operating system	Required for operating system **LogonUser** call.
Bypass traverse checking	Required for operating system **LogonUser** call.
Replace a process level token	Required for operating system **LogonUser** call.

Your domain user account must have these Windows user rights set as effective user rights as listed in the Local Security Policy application. If they are not, set them using either the Local Security Policy application locally on the server, or by using the Domain Security Application domain wide.

Parent topic: Using Active directory (Windows only)

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12220_

3.4.1.2. Changing the user name associated with WebSphere MQ Services

You might need to change the user name associated with WebSphere® MQ Services from MUSR_MQADMIN to something else. (For example, you might need to do this if your queue manager is associated with DB2®, which does not accept user names of more than 8 characters.)

To change the user name :

1. **Create a new user account (for example NEW_NAME)**
2. **Use the Prepare WebSphere MQ Wizard to enter the account details of the new user account.**
   Windows Vista and Windows Server 2008 users with UAC enabled only: Use the following command line to set the new account:

   ```
   AMQMSRVN -user <domain\>NEW_NAME -password <password>
   ```

   Where `NEW_NAME` is the new user name you have chosen. This can be qualified by a domain name if required. WebSphere MQ allocates the correct security rights and group membership to the new user account. Ensure the application is run with Administrator privileges.

If for any reason you need to reset the user account back to the default MUSR_MQADMIN account, use the following command:

```
AMQMJPSE -r
```

Parent topic: Using Active directory (Windows only)

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12230_

3.4.2. Controlling access (Windows only)

When you install WebSphere® MQ, default access permissions are set up for the AMQMSRVN process. These default access permissions grant access to the process to:

- mqm (local WebSphere MQ administrators group)
- Administrators (local administrators of this machine) (Windows only)

These permissions restrict access to the alert monitor taskbar application and the amqmdain command to these users and groups only. Other users trying to access these functions are denied access. The WebSphere MQ Explorer uses amqmdain to start and end queue managers. If the WebSphere MQ Explorer is run by a user not authorized to run amqmdain, then an error is reported when the user attempts to start or stop a queue manager.

Before users run the WebSphere MQ Explorer, you must configure the access permissions of the objects involved. Use a tool called *DCOMCNFG.EXE*, shipped with Windows systems, to do this.

Using DCOMCNFG.EXE to change access permissions

Parent topic: Security on Windows

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12240_

3.4.2.1. Using DCOMCNFG.EXE to change access permissions

To start DCOMCNFG.EXE:

1. Click Start.
2. Click Run.
3. Type `dcomcnfg` in the open input field.
4. Click OK. The Component Services Window opens.
5. Expand Component services.
6. Expand Computers.
7. Expand My Computer.
8. Click the DCOM Config node.

A list of applications is displayed. From this list:

1. Find and highlight the IBM® MQSeries® Services entry.
2. Right-click IBM MQSeries Services, and then click Properties.
   This displays information about the location of the DCOM server (AMQMSRVN.EXE), together with its identity and security properties.
3. Select the Security page to view or modify the launch, access, or configuration permissions.
4. Stop and restart the IBM MQSeries service from the Windows Services control panel for your changes to take effect. (If your changes affect a user who is currently logged on, that user must log off and on again). Do the following:
   a. Access the panel by clicking Start > Run > services.msc. A list of services is displayed.
   b. Right-click IBM MQSeries, and select Properties. The IBM MQSeries Properties pages is displayed.
   c. Click Stop. The IBM MQSeries service stops.
   d. Click Start. The IBM MQSeries service restarts.

In addition to being able to add to the list of users that are allowed access to a service, you can deny access to specific users and groups. This means that you can grant access to a group of users (by specifying a group name) but deny access to individuals within that group.

Parent topic: Controlling access (Windows only)

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12250_

3.4.3. Changing the password of the AMQMSRVN user account

If AMQMSRVN is running under the local user account MUSR_MQADMIN (or another local user account), you can change the password for the account as follows:

1. Stop the `IBM MQSeries` service from the Computer Management panel.
2. Close any WebSphere® MQ programs that are using the AMQMSRVN COM server (this includes the alert monitor, taskbar, and so on).
3. Change the MUSR_MQADMIN password in the same way that you would change an individual's password.
4. Use DCOMCNFG.EXE to bring up the properties pages for the `IBM MQSeries` service.
5. Select the Identity Page.
6. Modify the password given for the MUSR_MQADMIN user account.

If AMQMSRVN is running under a domain user account, you can also change the password for the account as follows:

1. Change the password for the domain account on the domain controller. You might need to ask your domain administrator to do this for you.
2. Use the Prepare WebSphere MQ Wizard to enter the account details including the new password.

The user account that AMQMSRVN runs under executes any MQSC commands that are issued by user interface applications, or performed automatically on system startup, shutdown, or service recovery. This user account must therefore have WebSphere MQ administration rights. By default it is added to the local mqm group on the server. If this membership is removed, the `IBM MQSeries` service will not work.

If a security problem arises with the DCOM configuration or with the user account that AMQMSRVN runs under, error messages and descriptions appear in the system event log. One common error is for a user not to have access or launch rights to the server. This error appears in the system log as a DCOM error with the following message description:

```
Access denied attempting to launch a DCOM server.  The server is:
{55B99860-F95E-11d1-ABB6-0004ACF79B59}
```

Parent topic: Security on Windows

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12260_

3.4.4. WebSphere MQ coordinating with DB2 as the resource manager

If you start your queue managers via the WebSphere MQ Explorer or are using WebSphere® MQ V7 and are having problems when coordinating DB2 check your queue manager error logs for an error like this:

```
23/09/2008 15:43:54 - Process(5508.1) User(MUSR_MQADMIN) Program(amqzxma0.exe)
AMQ7604: The XA resource manager 'DB2 MQBankDB database' was not available when called
      for xa_open. The queue manager is continuing without this resource manager.
```

Explanation: The user id (default name is MUSR_MQADMIN) which runs the MQ Services process `amqmsrvn.exe` is still running with an access token which does not contain group membership information for the group DB2USERS.

Solution: After you have ensured that the MQ Services user id is a member of DB2USERS, use the following sequence of commands:

- stop the MQ service (for example by issuing net stop "IBM MQSeries").
- stop any other processes running under the same user id.

- restart these processes.

Rebooting the machine would ensure the previous steps, but is not usually necessary.

Parent topic: Security on Windows

This build: January 26, 2011 11:54:44

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12205_

3.5. Extending the WebSphere MQ Explorer

This information applies to WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform only).

WebSphere MQ for Windows, and WebSphere MQ for Linux (x86 platform) provide an administration interface called the WebSphere MQ Explorer to perform administration tasks as an alternative to using control or MQSC commands. The WebSphere MQ Explorer presents information in a style consistent with that of the Eclipse framework and the other plug-in applications that Eclipse supports.

This chapter explains how the features, and perspectives, available with the WebSphere MQ Explorer can be extended to give further administrative control over your WebSphere MQ system. Through extending the WebSphere MQ Explorer system administrators have the ability to customize the WebSphere MQ Explorer to improve the way they administer WebSphere MQ.

Who this chapter is for

What you need to know to understand this chapter

Introduction

Writing an Eclipse plug-in for the WebSphere MQ Explorer

Applying plug-ins to the WebSphere MQ Explorer
You can either run a plug-in with WebSphere MQ Explorer from the Eclipse workbench, or apply updates from a plug-in to WebSphere MQ Explorer permanently.

Parent topic: Administration using the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12270_

3.5.1. Who this chapter is for

This chapter is for system administrators and system programmers who intend to extend, and customize, the functions that are provided by the WebSphere® MQ Explorer.

Parent topic: Extending the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12280_

3.5.2. What you need to know to understand this chapter

To use this chapter, you need the following:

- A good understanding of the Eclipse framework and of the utilities associated with it.
- Experience in writing Eclipse plug-ins.
  Information on how to write Eclipse plug-ins is available in the *Platform Plug-in Developers Guide*, found in Eclipse help.
  1. From the WebSphere® MQ Explorer toolbar click, Help -> Help Contents.

2. Click Platform Plug-in Developers Guide.

- A good understanding of the features available with the WebSphere MQ Explorer.

Parent topic: Extending the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12290_

3.5.3. Introduction

The WebSphere® MQ Explorer is Eclipse based, and as such acquires all its functionality, and perspective information, through various plug-ins supplied with WebSphere MQ. To extend the WebSphere MQ Explorer you are required to write one, or more, Eclipse plug-ins. Through writing a plug-in you can extend the function of the WebSphere MQ Explorer in the following ways:

- Add further menu options to existing context menus and associate actions with them.
- Add tree nodes to the navigation view, and associated content pages.

When writing a plug-in, you will need to supply:

The plugin.xml file

Use the plugin.xml file to specify *extension points*. Extension points provide the means by which plug-in developers can extend the functionality of the WebSphere MQ Explorer. There are many types extension point available within the WebSphere MQ Explorer and Eclipse. Each type of extension point is used to extend the Explorer in a different way. Most extension points are associated with a Java jar file. For more information on the extension points available, see Utilizing extension points.

Java jar file

Use Java jar files contain the classes that provide the code needed to implement the added functionality specified by the extension points declared in the plugin.xml file. Every Java jar file is associated with at least one extension point.

WebSphere MQ supplies sample Eclipse plug-ins called *simple*, and *menu*. The simple plug-in utilizes all the extension points provided in the WebSphere MQ Explorer to extend the Explorer in a number of basic ways. The simple plug-in can be used as a basis for writing your own Eclipse plug-ins. For instructions of how to import the simple plug-in, see Importing the sample Eclipse plug-ins.

Importing the sample Eclipse plug-ins
Instructions for importing the sample Eclipse plug-ins.

Parent topic: Extending the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12300_

3.5.3.1. Importing the sample Eclipse plug-ins

Instructions for importing the sample Eclipse plug-ins.

To import the sample Eclipse plug-ins, complete the following steps:

1. Open the WebSphere® MQ Explorer in an Eclipse workbench.
2. Open the Plug-in Development perspective.
3. Click File > Import to open the Import wizard.
   In the Import wizard, complete the following steps:
   a. Click Plug-in Development > Plug-ins and Fragments.
   b. Select the Projects with source folders check box and click Next.
   c. Select one or more from:
      com.ibm.mq.explorer.sample.simple
      com.ibm.mq.explorer.sample.menus
      com.ibm.mq.explorer.jmsadmin.sample.menus

com.ibm.mq.explorer.tests.sample

d. Click Finish

You have now imported the sample Eclipse plug-ins.
Parent topic: Introduction

Related concepts
Starting the WebSphere MQ Explorer in standalone mode or Eclipse workbench mode

This build: January 26, 2011 11:54:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12310_

3.5.4. Writing an Eclipse plug-in for the WebSphere MQ Explorer

This section details how to write an Eclipse plug-in for the WebSphere® MQ Explorer. It is assumed that you have the prerequisite knowledge detailed in What you need to know to understand this chapter.

To write an Eclipse plug-in for the WebSphere MQ Explorer, you must utilize the extension points available to extend the functionality of the WebSphere MQ Explorer. The most common extension points are described, and accompanied by a number of code extracts from the simple plug-in to provide basic implementation examples. You must import the simple plug-in if you want access to the code that it contains. For instructions on how to import the simple plug-in, see Importing the sample Eclipse plug-ins.

The environment in which the WebSphere MQ Explorer is extended is an event driven interface. For example, when a Register extension point is extended with an instance of a user-written class that extends the IExplorerNotify interface, the user-written class will be called back when an event occurs. For example, when a queue manager is created. Many of these notifications include a MQExtObject as one of their arguments. An MQExtObject relates to the WebSphere MQ object that caused the event. A user-written class can call any of the MQExtObject public methods to find out about the object.

The IExplorerNotify interface, the associated MQExtObject, and other external definitions are documented in the WebSphere MQ Explorer JavaDoc. For information on how to access the WebSphere MQ Explorer JavaDoc, see Accessing the WebSphere MQ Explorer Javadoc.

Accessing the WebSphere MQ Explorer Javadoc
External definitions relating to the WebSphere MQ Explorer are documented in Javadoc and can be displayed as HTML pages.

Utilizing extension points
Instructions on how to use extension points.

Parent topic: Extending the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12320_

3.5.4.1. Accessing the WebSphere MQ Explorer Javadoc

External definitions relating to the WebSphere MQ Explorer are documented in Javadoc and can be displayed as HTML pages.

To access the WebSphere® MQ Explorer Javadoc as HTML pages, complete the following steps:

1. Open the WebSphere MQ Explorer.
2. Click Help > Help Contents.
3. Expand WebSphere MQ Explorer.
4. Click API Reference.

The WebSphere MQ Explorer Javadoc is displayed in HTML.
Parent topic: Writing an Eclipse plug-in for the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12330_

3.5.4.2. Utilizing extension points

Instructions on how to use extension points.

This topic describes how to implement the extension points available in Eclipse plug-ins for the WebSphere® MQ Explorer.

For further information on using extension points see the WebSphere MQ Explorer help, as follows:

1. **Click Help->Help Contents.**
2. **Expand Platform Plug-in Developers Guide.**
3. **Click Programmers Guide.**

For information on how to include an extension point, see Plugging into the workbench->Basic workbench extension points in the Programmers Guide.

Through utilizing the available extension points, you can extend the function of the WebSphere MQ Explorer in the following ways:

- **Register extension points.**
- **Add further menu options to existing context menus and associate actions with them.**
- **Add tree nodes to the navigation view and associate content pages with them.**
- **Add property tabs to property dialogs and associate property pages with them.**

Multiple extension points of the same type can be included in a single plug-in. The extension points that you use will be dependent on the way in which you intend to extend the functionality of the WebSphere MQ Explorer. However, every plug-in for the WebSphere MQ Explorer must use the *register* extension point. For details on the register extension point, see Register.

Register

Add tree node

Add content page

Add context menu item

Adding a property tab to an Eclipse property dialog
A *property tab* extension point is used to add a property tab to a property dialog and an associated property page.

Parent topic: Writing an Eclipse plug-in for the WebSphere MQ Explorer

This build: January 26, 2011 11:54:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12340_

3.5.4.2.1. Register

The *register* extension point is used for the following:

- **To allow your plug-in to register itself with the WebSphere® MQ Explorer. Every plug-in for the WebSphere MQ Explorer must include this extension point in plugin.xml. With out it, any function your plug-in adds to the WebSphere MQ Explorer will not be activated.**
- **To enable notify events. For information on notify events, see Notify events.**

The following code extract is taken from the file, plugin.xml, from the simple plug-in and shows a basic implementation of the register extension point:

```
<extension
        id="com.ibm.mq.explorer.sample.simple"
        name="Simple Sample"
        point="com.ibm.mq.explorer.ui.registerplugin">
      <pluginDetails
            pluginId="com.ibm.mq.explorer.sample.simple"
```

```
            name="Simple"
            class="com.ibm.mq.explorer.sample.simple.SimpleNotify"
            enabledByDefault="true"
            description="a very simple sample plugin to Explorer"
            vendor="IBM">
      </pluginDetails>
</extension>
```

Enabling and disabling a plug-in

Notify events

Parent topic: Utilizing extension points

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12350_

3.5.4.2.1.1. Enabling and disabling a plug-in

All plug-ins that contain the register extension point can be enabled, or disabled, within the WebSphere® MQ Explorer by doing the following:

1. **From the WebSphere MQ Explorer toolbar click, Window -> Preferences.**
2. **Expand IBM® WebSphere MQ.**
3. **Click Enable plug-ins.**
   All registered plug-ins are displayed.
4. **Select all plug-ins that should be enabled.**
5. **Click OK.**

Parent topic: Register

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12360_

3.5.4.2.1.2. Notify events

Within the WebSphere® MQ Explorer, when a WebSphere MQ object is created, or manipulated, a java object relating to the WebSphere MQ object can be generated. These Java objects can be used to find the name, type, and other externalized attributes of a WebSphere MQ object.

For Java objects to be generated, the register extension point must specify a class. In the plugin.xml file from the simple plug-in, the class specified is as follows:

```
class="com.ibm.mq.explorer.sample.simple.SimpleNotify"
```

This class contains a number of object specific methods. When a WebSphere MQ object is created, or manipulated, the appropriate method from the notify class is called. This class can be used as a basis for writing your own class. For the methods that this class must contain refer to the WebSphere MQ Explorer JavaDoc. For information on how to access the WebSphere MQ Explorer JavaDoc, see Accessing the WebSphere MQ Explorer Javadoc.
Parent topic: Register

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12370_

3.5.4.2.2. Add tree node

A *tree node* extension point is used to add a tree node to the navigation view and associate it with a content page.

The following code extract is taken from the file, plugin.xml, from the simple plug-in and shows a basic implementation of the tree node extension point:

```
<extension
```

```
          id="com.ibm.mq.explorer.samples.simpleTreeNode"
          name="Simple TreeNode"
          point="com.ibm.mq.explorer.ui.addtreenode">
       <treeNode
             pluginId="com.ibm.mq.explorer.sample.simple"
             name="com.ibm.mq.explorer.sample.simple"
             class="com.ibm.mq.explorer.sample.simple.SimpleTreeNodeFactory"
             treeNodeId="com.ibm.mq.explorer.sample.simple"
             sequence="888">
       </treeNode>
</extension>
```

As well as declaring the tree node extension point in plugin.xml, the following classes are needed:

- A class that contains a method that checks the id of any incoming tree node to determine whether to add sub nodes to it. This class must implement com.ibm.mq.explorer.ui.extensions.ITreeNodeFactory, and IExecutableExtension. For the methods that this class must contain refer to the WebSphere® MQ Explorer JavaDoc. For information on how to access the WebSphere MQ Explorer JavaDoc, see Accessing the WebSphere MQ Explorer Javadoc.
  A working example of this class is available in the simple plug-in, called SimpleTreeNodeFactory.java
- A class that contains methods that return information about any new tree nodes, such as the name, id, and the associated content page class. This class must extend com.ibm.mq.ui.extensions.TreeNode. For the methods that this class must contain refer to the WebSphere MQ Explorer JavaDoc.
  A working example of this class is available in the simple plug-in, called SimpleTreeNode.java.

Parent topic: Utilizing extension points

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12380_

3.5.4.2.3. Add content page

A *content page* extension point is used to add a content page to the content view. A content page can be associated with a tree node.

The following code extract is taken from the file, plugin.xml, from the simple plug-in and shows a basic implementation of the content page extension point:

```
<extension
        id="com.ibm.mq.explorer.sample.simpleContentPage"
        name="Simple ContentPage"
        point="com.ibm.mq.explorer.ui.addcontentpage">
      <contentPage
            pluginId="com.ibm.mq.explorer.sample.simple"
            name="com.ibm.mq.explorer.sample.simple"
            class="com.ibm.mq.explorer.sample.simple.SimpleContentPageFactory"
            contentPageId="com.ibm.mq.explorer.sample.simple">
      </contentPage>
</extension>
```

As well as declaring the content page extension point in plugin.xml, the following classes are needed:

- A class that contains methods that perform a number of functions such as return the content page id, create the content page, and set the object to draw the page. This class must extend com.ibm.mq.ui.extensions.ContentsPage. The class com.ibm.mq.explorer.ui.extensions.ContentTitleBar can be used to create a title for the content page consistent with the other content pages in the WebSphere® MQ Explorer. For the methods that this class must contain refer to the WebSphere MQ Explorer JavaDoc. For information on how to access the WebSphere MQ Explorer JavaDoc, see Accessing the WebSphere MQ Explorer Javadoc.
  A working example of this class is available in the simple plug-in, called SimpleContentPage.java.
- A class that contains a method that returns an instance of the class extending ContentPage. This class must implement com.ibm.mq.explorer.ui.extensions.IContentPageFactory, and IExecutableExtension. For the methods that this class must contain refer to the WebSphere MQ Explorer JavaDoc.
  A working example of this class is available in the simple plug-in, called SimpleContentPageFactory.java

Parent topic: Utilizing extension points

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12390_

3.5.4.2.4. Add context menu item

A *context menu* extension point is used to add context menu items to the WebSphere® MQ Explorer.

The following code extract is taken from the file, plugin.xml, from the simple plug-in and shows a basic implementation of the context menu extension point:

```
<extension
        id="com.ibm.mq.explorer.sample.simple.object1"
        name="Object1"
        point="org.eclipse.ui.popupMenus">
      <objectContribution
             objectClass="com.ibm.mq.explorer.ui.extensions.MQExtObject"
             id="com.ibm.mq.explorer.sample.simple.obj1">
           <visibility>
                <and>
                     <pluginState
                       value="activated"
                       id="com.ibm.mq.explorer.ui">
                     </pluginState>
                     <objectClass
                       name="com.ibm.mq.explorer.ui.extensions.MQExtObject">
                     </objectClass>
                     <objectState
                       name="PluginEnabled"
                       value="com.ibm.mq.explorer.sample.simple">
                     </objectState>
                </and>
           </visibility>
           <action
             label="Simple: Sample action on any MQExtObject"
             class="com.ibm.mq.explorer.sample.simple.MenuActions"
             menubarPath="additions"
             id="com.ibm.mq.explorer.sample.simple.obj.action1">
           </action>
      </objectContribution>
</extension>
```

Additional context menu items are added using the Eclipse extension point `org.eclipse.ui.popupMenus`. The `<visibility>` attribute in the above extract contains the elements that control the conditions under which the context menu item is displayed. These conditions include tests on the plug-in state, the type of object, and the state of the object. For example, a content menu item can be displayed for local queues only, or for remote queue managers only.

Parent topic: Utilizing extension points

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12400_

3.5.4.2.5. Adding a property tab to an Eclipse property dialog

A *property tab* extension point is used to add a property tab to a property dialog and an associated property page.

The following code extract is taken from the file plugin.xml, from the simple plug-in, and shows a basic implementation of the property tab extension point:

```
<extension
        id="com.ibm.mq.explorer.samples.simplePropertyTab"
        name="Simple Property Tab"
        point="com.ibm.mq.explorer.ui.addpropertytab">
      <propertyTab
           class="com.ibm.mq.explorer.sample.simple.SimplePropertyTabFactory"
           objectId="com.ibm.mq.explorer.queuemanager"
           pluginId="com.ibm.mq.explorer.sample.simple"
           name="com.ibm.mq.explorer.sample.simple"
           propertyTabId="com.ibm.mq.explorer.sample.simple.propertyTab"
           propertyTabName="Simple Sample Property Tab"/>
</extension>
```

As well as declaring the property tab extension point in plugin.xml, the following classes are needed:

- **A class that contains a method that creates and returns a property page to be displayed when a user clicks the**

property tab. This class must implement com.ibm.mq.explorer.ui.extensions.IPropertyTabFactory. For the methods that this class must contain refer to the WebSphere® MQ Explorer JavaDoc. For information on how to access the WebSphere MQ Explorer JavaDoc, see Accessing the WebSphere MQ Explorer Javadoc.
A working example of this class, called SimplePropertyTabFactory.java, is available in the simple plug-in.

- A class used for creating the property page must extend com.ibm.mq.ui.extensions.PropertyPage. For the methods that this class must contain refer to the WebSphere MQ Explorer JavaDoc.
  A working example of this class, called SimplePropertyPage.java, is available in the simple plug-in.

Parent topic: Utilizing extension points

This build: January 26, 2011 11:56:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22780_

3.5.5. Applying plug-ins to the WebSphere MQ Explorer

You can either run a plug-in with WebSphere® MQ Explorer from the Eclipse workbench, or apply updates from a plug-in to WebSphere MQ Explorer permanently.

To run plug-ins with WebSphere MQ Explorer from the Eclipse workbench, complete the following steps:

1. Select the plug-in from the Package Explorer.
2. Click Run > Run As > Eclipse Application.
   A new Eclipse workbench opens.
3. In the new Eclipse workbench, open the WebSphere MQ Explorer perspective.
4. In the Explorer preferences section, select the "Enable plug-ins" page and enable the relevant sample plug-in or plug-ins.

To permanently apply updates to the WebSphere MQ Explorer provided by a plug-in, complete the following steps:

1. With a file browser, find the plug-in file that will provide the functionality extensions to the WebSphere MQ Explorer.
2. Copy the plug-in file, and paste it into `C:\Program Files\IBM\WebSphere MQ\eclipse\plugins`, (or equivalent on Linux (x86 platform)).
3. Restart the WebSphere MQ Explorer.

The updates provided by the plug-in are applied to the WebSphere MQ Explorer.
Parent topic: Extending the WebSphere MQ Explorer

This build: January 26, 2011 11:54:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12410_

4. Using the WebSphere MQ Taskbar application (Windows only)

The WebSphere® MQ Taskbar application displays an icon in the Windows system tray on the server. The icon provides you with the current status of WebSphere MQ and a context menu from which you can perform some simple actions.

On Windows, the WebSphere MQ icon is in the system tray on the server and is overlaid with a color-coded status symbol, which can have one of the following meanings:

Green

Healthy; no alerts at present

Blue

Indeterminate; WebSphere MQ is starting up or shutting down

Yellow

Alert; one or more services are failing or have already failed

To display the context menu, right-click the WebSphere MQ icon. From the context menu you can perform the following actions:

- Click Open to open the WebSphere MQ Alert Monitor
- Click Exit to exit the WebSphere MQ Taskbar application
- Click WebSphere MQ Explorer to start the WebSphere MQ Explorer

- Click Stop WebSphere MQ to stop WebSphere MQ
- Click About WebSphere MQ to display information about the WebSphere MQ Alert Monitor

Using the WebSphere MQ alert monitor application (Windows only)

Parent topic: System Administration Guide

This build: January 26, 2011 11:54:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12170_

4.1. Using the WebSphere MQ alert monitor application (Windows only)

The WebSphere® MQ alert monitor is an error detection tool that identifies and records problems with WebSphere MQ on a local machine. The alert monitor displays information about the current status of the local installation of a WebSphere MQ server It also monitors the Windows Advanced Configuration and Power Interface (ACPI) and ensures the ACPI settings are enforced.

From the WebSphere MQ alert monitor, you can:

- Access the WebSphere MQ Explorer directly
- View information relating to all outstanding alerts
- Shut down the WebSphere MQ service on the local machine
- Route alert messages over the network to a configurable user account, or to a Windows workstation or server

Parent topic: Using the WebSphere MQ Taskbar application (Windows only)

This build: January 26, 2011 11:54:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12180_

5. WebSphere MQ control commands

How to use WebSphere MQ control commands

The control commands
This collection of topics provides reference information for each of the WebSphere® MQ control commands. These control commands require that the ID is in the mqm group.

Comparing command sets

Managing keys and certificates

Parent topic: System Administration Guide

This build: January 26, 2011 11:55:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15490_

5.1. How to use WebSphere MQ control commands

This chapter describes how to use the WebSphere® MQ control commands. If you want to issue control commands, your user ID must be a member of the mqm group. For more information about this, see Authority to administer WebSphere MQ on UNIX and Windows systems. In addition, note the following environment-specific information:

WebSphere MQ for Windows

All control commands can be issued from a command line. Command names and their flags are not case sensitive: you can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However,

arguments to control commands (such as queue names) are case sensitive.

In the syntax descriptions, the hyphen (-) is used as a flag indicator. You can use the forward slash (/) instead of the hyphen.

WebSphere MQ for UNIX systems

All WebSphere MQ control commands can be issued from a shell. All commands are case-sensitive.

A subset of the control commands can be issued using the WebSphere MQ Explorer.

Names of WebSphere MQ objects

Syntax diagrams
The syntax for a command and its options is presented in the form of a diagram. There are two types of syntax diagram: railroad diagrams and dotted decimal diagrams.

Example syntax diagram

Syntax help

Parent topic: WebSphere MQ control commands

This build: January 26, 2011 11:55:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15500_

5.1.1. Names of WebSphere MQ objects

In general, the names of WebSphere® MQ objects can have up to 48 characters. This rule applies to all the following objects:

- Queue managers
- Queues
- Process definitions
- Namelists
- Clusters
- Listeners
- Services
- Authentication information objects
- Topics

The maximum length of channel, and client connection channel names is 20 characters.Topic strings can be 10240 bytes.

The characters that can be used for all WebSphere MQ names are:

- Uppercase A–Z
- Lowercase a–z
- Numerics 0–9
- Period (.)
- Underscore (_)
- Forward slash (/) (see note 1)
- Percent sign (%) (see note 1)

Note:

1. Forward slash and percent are special characters. If you use either of these characters in a name, the name must be enclosed in double quotation marks whenever it is used.
2. Leading or embedded blanks are not allowed.
3. National language characters are not allowed.
4. Names can be enclosed in double quotation marks, but this is essential only if special characters are included in the name.

Parent topic: How to use WebSphere MQ control commands

This build: January 26, 2011 11:55:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15510_

5.1.2. Syntax diagrams

The syntax for a command and its options is presented in the form of a diagram. There are two types of syntax diagram: railroad diagrams and dotted decimal diagrams.

Both types of syntax diagram tell you what options you can supply with the command, and how to enter them, and indicate relationships between different options, and sometimes different values of an option.

- **Railroad diagrams are a visual format suitable for sighted users.**
- **Dotted decimal diagrams are text-based diagrams that are helpful for blind or partially-sighted users, and work well with screen-reader software.**

How to read railroad diagrams
Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

How to read dotted decimal diagrams
In dotted decimal format, each syntax element is written on a separate line, starting with a dotted decimal number.

Parent topic: How to use WebSphere MQ control commands

This build: January 26, 2011 11:56:25

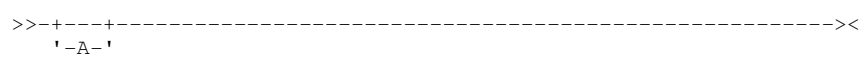
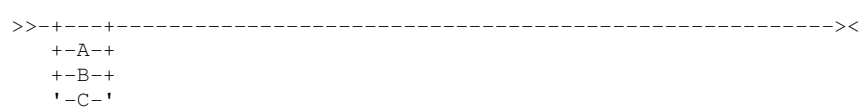
© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
mq50060_

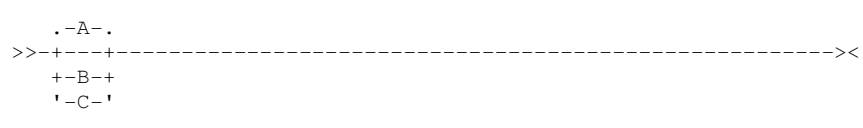
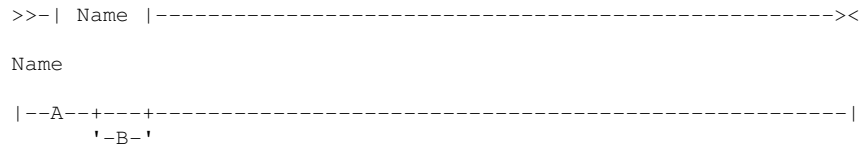
5.1.2.1. How to read railroad diagrams

Each railroad diagram begins with a double right arrow and ends with a right and left arrow pair. Lines beginning with a single right arrow are continuation lines. You read a railroad diagram from left to right and from top to bottom, following the direction of the arrows.

Other conventions used in railroad diagrams are:

Table 1. How to read railroad diagrams

Convention	Meaning
	You must specify values A, B, and C. Required values are shown on the main line of a railroad diagram.
	You may specify value A. Optional values are shown below the main line of a railroad diagram.
	Values A, B, and C are alternatives, one of which you must specify.
	Values A, B, and C are alternatives, one of which you might specify.

	You might specify one or more of the values A, B, and C. Any required separator for multiple or repeated values (in this example, the comma (,)) is shown on the arrow.
	You might specify value A multiple times. The separator in this example is optional.
	Values A, B, and C are alternatives, one of which you might specify. If you specify none of the values shown, the default A (the value shown above the main line) is used.
	The railroad fragment Name is shown separately from the main railroad diagram.
Punctuation and uppercase values	Specify exactly as shown.

Parent topic: Syntax diagrams

This build: January 26, 2011 11:56:25

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
mq50063_

5.1.2.2. How to read dotted decimal diagrams

In dotted decimal format, each syntax element is written on a separate line, starting with a dotted decimal number.

If two or more syntax elements are always present together (or always absent together), they can appear on the same line, as they can be considered as a single compound syntax element.

Each line starts with a dotted decimal number, for example `3` or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that your screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number `3.1`) are mutually exclusive alternatives. For example, if you hear the lines `3.1 USERID`, `3.1 SYSTEMID`, you know that your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with the dotted decimal number `3` is followed by a series of syntax elements with the dotted decimal number `3.1`, all the syntax elements numbered `3.1` are subordinate to the syntax element numbered `3`.

Characters such as commas that are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. They might appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line might also show another symbol giving information about the syntax elements; all these symbols are explained below. For example, the lines `5.1* ,`, `5.1 LASTRUN`, `5.1 DELETE` mean that if you use more than one of the syntax elements `LASTRUN` and `DELETE`, they must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the `%` symbol, this indicates a reference that is defined elsewhere. The string following the `%` is the name of a syntax fragment, rather than a literal. For example, the line `2.1 %OP1` means that at this point, you should refer to the separate syntax fragment `OP1`. `OP1`, in the syntax from which this example was taken, gave a list of further options.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the escape character, which is a \ (backslash). For example, the `*` symbol can be used next to a dotted decimal number to mean that this syntax element can be repeated. If a syntax element actually starts with the `*` symbol, for example a syntax element `* FILE` with the dotted decimal number 3, it is given in the format `3 \* FILE`. If the format is `3* FILE`, this means that there is a syntax element `FILE`, which can be repeated. If the format was `3* \* FILE`, this means that there is a syntax element `* FILE`, which can be repeated.

The words and symbols used next to the dotted decimal numbers are as follows:

- `?` means an optional syntax element. If a dotted decimal number is followed by the `?` symbol, this means that all the syntax elements with that dotted decimal number, and any subordinate syntax elements that they each have, are optional. If there is only one syntax element with that dotted decimal number, the `?` symbol appears on the same line as the syntax element, for example `5? NOTIFY`. If there is more than one syntax element with that dotted decimal number, the `?` symbol appears on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines `5 ?`, `5 NOTIFY`, `5 UPDATE`, you know that the syntax elements `NOTIFY` and `UPDATE` are optional; you can choose one or none of them. The `?` symbol is equivalent to a bypass line in a railroad diagram.
- `!` means a default syntax element. If a dotted decimal number is followed by the `!` symbol, appended to the last digit of the dotted decimal number, this means that this syntax element is the default of all the elements with the same dotted decimal number. Only one of the syntax elements that share the same dotted decimal number can specify a `!`. For example, if you hear the lines `2? FILE`, `2.1! (KEEP)`, `2.1 (DELETE)`, you know that `(KEEP)` is the default option for the `FILE` keyword. If you include the `FILE` keyword but do not state your choice of option, the default option `KEEP` is applied. As well as the particular syntax element marked with the `!` symbol, the default also applies to the next higher dotted decimal number. In the example above, the default applies to `2? FILE` as well as to `2.1! (KEEP)`, meaning that if you omit the word `FILE`, the default `FILE (KEEP)` is used. However, you might instead hear the lines `2? FILE`, `2.1`, `2.1.1! (KEEP)`, `2.1.1 (DELETE)`. As the default only applies to the next higher dotted decimal number, which in this case is `2.1`, it does not apply to `2? FILE`. In this case, if you omit the word `FILE`, nothing is used.
- `*` means a syntax element that is optional and can be repeated. If a dotted decimal number is followed by the `*` symbol, this means that this syntax element it is optional, and can be repeated. For example, if you hear the line `5.1* data-area`, you know that you can include more than one data area, or you can include none. If you hear the lines `3*`, `3 HOST`, `3 STATE`, you know that you can include `HOST`, `STATE`, both, or nothing. Note that if a dotted decimal number has an asterisk next to it, and there is only one item with that dotted decimal number, you can repeat that same item more than once. If a dotted decimal number has an asterisk next to it, and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the example above, you could write `HOST STATE`, but you could not write `HOST HOST`. The `*` symbol is equivalent to a loop back line in a railroad syntax diagram.
- `+` means a syntax element that must be included at least once, and can be repeated. If a dotted decimal number is followed by the `+` symbol, this means that this syntax element must be included at least once, and can be repeated. For example, if you hear the line `6.1+ data-area`, you know that you must include at least one data area, and you can include more than one. If you hear the lines `2+`, `2 HOST`, `2 STATE`, you know that you must include `HOST`, `STATE`, or both. As for the `+` symbol, you can only repeat a particular item if it is the only item with that dotted decimal number. The `+` symbol, like the `*` symbol, is equivalent to a loop back line in a railroad syntax diagram.

Parent topic: Syntax diagrams

This build: January 26, 2011 11:56:25

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
mq50066_

5.1.3. Example syntax diagram

Here is an example syntax diagram that describes the hello command:
Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
+ 'Hello Command\n'
+ '\n'
+ '>>-hello--+----------+--+--------------+----------------------><\n'
```

```
+ '           \'-| Name |-\'   \'-| Greeting |-\'    \n'
+ '\n'
+ 'Name\n'
+ '\n'
+ '    .-,-----------.    \n'
+ '    V           (1) |    \n'
+ '|---- --name-----+-----------------------------------------|\n'
+ '\n'
+ 'Greeting\n'
+ '\n'
+ '|--, --how are you?-------------------------------------|\n'
+ '\n'
+ '
');} //]]>
```

Notes:

1. **You can code up to three names.**

According to the syntax diagram, these are all valid versions of the hello command:

```
hello
hello name
hello name, name
hello name, name, name
hello, how are you?
hello name, how are you?
hello name, name, how are you?
hello name, name, name, how are you?
```

The space before the *name* value is significant, and that if you do not code *name* at all, you must still code the comma before `how are you?`.

Parent topic: How to use WebSphere MQ control commands

This build: January 26, 2011 11:55:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15520_

5.1.4. Syntax help

You can obtain help for the syntax of any control command by entering the command followed by a question mark. WebSphere® MQ responds by listing the syntax required for the selected command.

The syntax shows all the parameters and variables associated with the command. Different forms of parentheses are used to indicate whether a parameter is required. For example:

```
CmdName [-x OptParam ] ( -c | -b ) argument
```

where:

`CmdName`

Is the command name for which you have requested help.

`[-x OptParam ]`

Square brackets enclose one or more optional parameters. Where square brackets enclose multiple parameters, you can select no more than one of them.

`( -c | -b )`

Brackets enclose multiple values, one of which you must select. In this example, you must select either flag c or flag b.

`argument`

A mandatory argument.

Examples

Parent topic: How to use WebSphere MQ control commands

This build: January 26, 2011 11:55:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15530_

5.1.4.1. Examples

1. Result of entering `endmqm ?`

```
endmqm [-z][-c | -w | -i | -p] QMgrName
```

2. Result of entering `rcdmqimg ?`

```
rcdmqimg [-z] [-m QMgrName] -t ObjType [GenericObjName]
```

Parent topic: Syntax help

This build: January 26, 2011 11:55:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15540_

5.2. The control commands

This collection of topics provides reference information for each of the WebSphere® MQ control commands. These control commands require that the ID is in the mqm group.

»addmqinf«
Add WebSphere MQ configuration information (»Windows and «UNIX platforms only).

amqccert
Check for incomplete certificate chains (Windows only).

amqmdain
Configure or control WebSphere MQ services control (Windows only).

»amqmfsck (file system check)«
Check that a file system is compliant with POSIX standards and capable of sharing queue manager data to support multi-instance queue managers.

amqtcert
Migrate certificates from WebSphere MQ 5.3 or 5.3.1 (Windows only).

crtmqcvx
Create data conversion code from data type structures.

crtmqm
Create a queue manager.

dltmqm
Delete a queue manager.

dmpmqaut
Dump a list of current authorizations for a range of WebSphere MQ object types and profiles.

dmpmqlog
Display and format a portion of the WebSphere MQ system log.

dspmq
Display information about queue managers.

dspmqaut
dspmqaut displays the authorizations of a specific WebSphere MQ object.

dspmqcsv
Display the status of a command server

dspmqfls
Display the file names corresponding to WebSphere MQ objects.

dspmqinf
Display WebSphere MQ configuration information (Windows and Unix platforms only).

dspmqrte
Determine the route that a message has taken through a queue manager network.

dspmqtrc
Format and display WebSphere MQ trace (Unix platforms only).

dspmqtrn
Display in-doubt transactions.

dspmqver
Display WebSphere MQ version and build information.

endmqcsv
Stop the command server for a queue manager.

endmqlsr
End all listener process for a queue manager.

endmqdnm
Stop the .NET monitor for a queue (Windows only).

endmqm
Stop a queue manager or switch to a standby queue manager.

endmqtrc
End trace for some or all of the entities that are being traced.

migmbbrk
The migmbbrk command migrates publish/subscribe configuration data from WebSphere Event Broker Version 6.0 or WebSphere Message Broker Version 6.0 or 6.1 to WebSphere MQ Version 7.0.1 or later versions.

mqftapp
Start the graphical interface to File Transfer (Windows and Linux x86 platforms only).

mqftrcv
Process files received on a server using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).

mqftrcvc
Process files received on a client (Windows and Linux x86 platforms only).

mqftsnd
Send a file from a server using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).

mqftsndc
Send a file from a client using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).

rcdmqimg
Write the image of an object or group of objects to the log for media recovery.

rcrmqobj
Recreate an object, or group of objects, from their images contained in the log.

rmvmqinf
Remove WebSphere MQ configuration information (Windows and Unix platforms only).

rsvmqtrn
Resolve in-doubt transactions

runmqchi
Run a channel initiator process to automate starting channels.

runmqchl

Start a sender or requester channel

runmqdlq
Start the dead-letter queue handler to monitor and process messages on the dead-letter queue.

runmqdnm
Start processing messages on a queue using the .NET monitor (Windows only).

runmqlsr
Run a listener process to listen for remote requests on various communication protocols.

runmqsc
Run WebSphere MQ commands on a queue manager.

runmqtmc
Start the trigger monitor on a client.

runmqtrm
Start the trigger monitor on a server.

setmqaut
Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

setmqcrl
Administer CRL (certificate revocation list) LDAP definitions in an Active Directory (Windows only).

setmqprd
Enroll a WebSphere MQ production license.

setmqscp
Publish client connection channel definitions in an Active Directory (Windows only).

strmqcfg
Start the WebSphere MQ Explorer (Windows and Linux x86 platforms only).

strmqcsv
Start the command server for a queue manager.

strmqm
Start a queue manager» or ready it for standby operation«.

strmqtrc
Enable trace at a specified level of detail, or report the level of tracing in effect.

Parent topic: WebSphere MQ control commands

This build: January 26, 2011 11:55:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15550_

5.2.1. addmqinf

Add WebSphere® MQ configuration information (»Windows and «UNIX platforms only).

Purpose
Use the addmqinf command to add information to the WebSphere MQ configuration data.

For example use dspmqinf and addmqinf to copy configuration data from the system where a queue manager was created, to other systems where the same »multi-instance« queue manager is also to be started.

Syntax
Syntax diagram format ○ Railroad diagram ○ Dotted decimal
Attribute= *Value*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '                                        .------------------------.    \n'
+ '               .- -s -QueueManager-.   V                          |    \n'
+ '>>-addmqinf -+--------------------+---- -v - Attribute= Value-+-><\n'
+ '              \'- -s -StanzaType---\'                                   \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-v *Attribute=Value*

The name and value of the stanza attributes to be placed in the stanza specified in the command. Table 1 lists the QueueManager stanza attribute values. The queue manager stanza is the only stanza that is currently supported.

Table 1. QueueManager stanza attributes

Attribute	*Value*	Required or optional
Name	The name of the queue manager. You must provide a different name from any other queue manager stanza on the system.	Required
Prefix	The directory path *under* which this queue manager data directory is stored by default. You can use **Prefix** to modify the location of the queue manager data directories. The value of **Directory** *is* automatically appended to this path.	Required
Directory	The name of the queue manager data directory. Sometimes the name must be provided (as in Example), because it is different to the queue manager name. Copy the directory name from the value returned by **dspmqinf**. The rules for transforming queue manager names into directory names are described in Queue manager name transformation.	Required
DataPath	The directory path where the queue manager data files are placed. The value of **Directory** is *not* automatically appended to this path - you must provide the transformed queue manager name as part of **DataPath**. If the **DataPath** attribute is omitted on UNIX, the queue manager data directory path is defined as ***Prefix*/*Directory***.	UNIX: Optional Windows: Required

Optional parameters

-s *StanzaType*

A stanza of the type *StanzaType* is added to the WebSphere MQ configuration.
The default value of *StanzaType* is `QueueManager`.
The only supported value of *StanzaType* is `QueueManager`.

Return codes

0	Successful operation
1	Queue manager location is invalid (either **Prefix** or **DataPath**)
39	Bad command-line parameters
45	Stanza already exists
46	Required configuration attribute is missing
69	Storage is not available
71	Unexpected error
72	Queue manager name error
100	Log location is invalid

Example

```
addmqinf -v DataPath=/MQHA/qmgrs/QM!NAME
  -v Prefix=/var/mqm
  -v Directory=QM!NAME
  -v Name=QM.NAME
```

Creates the following stanza in mqs.ini:

```
QueueManager:
    Name=QM.NAME
    Prefix=/var/mqm
    Directory=QM!NAME
    DataPath=/MQHA/qmgrs/QM!NAME
```

Usage notes

Use dspmqinf with addmqinf to create an instance of a »multi-instance« queue manager on a different server.

To use this command you must be a WebSphere MQ administrator and a member of the `mqm` group.

Related commands

dspmqinf	Display WebSphere MQ configuration information
rmvmqinf	Remove WebSphere MQ configuration information

Parent topic: The control commands

This build: January 26, 2011 11:55:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15702_

5.2.2. amqccert

»Check for incomplete certificate chains (Windows only).«

Purpose

The amqccert command applies to WebSphere® MQ for Windows only.

The amqccert command is used during SSL Certificate Migration from WebSphere MQ for Windows Version 5.3, or Version 5.3.1. SSL Certificate Migration instructions are detailed in the WebSphere MQ Migration Information.

In this section when referring to a WebSphere MQ Certificate Store file, we are specifically referring to a WebSphere MQ for Windows Version 5.3, or Version 5.3.1, Certificate Store file.

To use amqccert you must be either an administrator or a member of the mqm group.

The amqccert control command is used to determine whether there are any incomplete certificate chains in a WebSphere MQ Certificate Store file. A report is generated that lists each incomplete certificate chain accompanied by information relating to the certificate chain.

Incomplete certificate chains must be completed before the SSL Certificate Migration process can continue. The following are available with WebSphere MQ for Windows Version 5.3, and Version 5.3.1, to help complete certificate chains:

- **The control command amqmcert (manage certificates).**
- **The Services snap-in.**

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

');break; default:document.write('

```
\n'
+ '>>-amqccert--File name----------------------------------------><\n'
+ '\n'
+ '
```

');} //]]>

Required parameters

File name

Specifies is the absolute (rather than relative) directory path name and file name (excluding the .sto suffix) of a WebSphere MQ Certificate Store.

Examples

In the following example reports the term, Microsoft Certificate Store, refers to a WebSphere MQ Certificate Store file.

```
amqccert C:\SSL\Client
```

Generates a report that details whether there are any incomplete certificate chains.

The following is an example of a report that details no incomplete certificate chains:

```
C:\ssl\client
5724-B41 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
The number of certificates in the Microsoft Certificate Store
                                                'c:\ssl\client' is '13'.

Certificate chain checking has completed with no failures.
The Check Certificate Chains (amqccert) command has completed.
```

The following is an example of a report the details two incomplete certificate chains:

```
C:\ssl\client
5724-B41 (C) Copyright IBM Corp. 1994, 2005.  ALL RIGHTS RESERVED.
The number of certificates in the Microsoft Certificate Store
                                                'c:\ssl\client' is '13'.

The signer certificate 'GlobalSign Primary Class 1 CA' is missing for
                                                 the following certificate.
Microsoft Certificate Store: 'c:\ssl\client'.
Certificate Subject:          'GlobalSign PersonalSign Class 1 CA'.
Certificate Issuer:           'GlobalSign Primary Class 1 CA'.
Certificate Serial Number:   '0400 0000 0000 FA3D EEE9 D9'.
Certificate Valid From:      '22/01/2004' to '28/01/2009'.

The signer certificate 'GlobalSign PersonalSign Class 1 CA' is missing
                                              for the following certificate.
Microsoft Certificate Store: 'c:\ssl\client'.
Certificate Subject:          'wm.shakespeare@hamlet.com'.
Certificate Issuer:           'GlobalSign PersonalSign Class 1 CA'.
Certificate Serial Number:   '0100 0000 0001 0170 978B 1E'.
Certificate Valid From:      '14/01/2005' to '14/02/2005'.

Certificate chain checking has completed with some failures.
The Check Certificate Chains (amqccert) command has completed.
```

Return codes

1	**amqccert** command usage error
2	User not authorized to run **amqccert** command
3	WebSphere MQ Certificate Store file not found
4	WebSphere MQ Certificate Store file is empty
5	WebSphere MQ Certificate Store file cannot be opened
6	No memory to allocate tables for storing root/intermediate certificates
7	Certificate is either an orphan or has expired
8	Windows operation failed

Related commands

amqtcert	Transfer certificates

Parent topic: The control commands

This build: January 26, 2011 11:55:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15560_

5.2.3. amqmdain

Configure or control WebSphere MQ services control (Windows only).

Purpose

The amqmdain command applies to WebSphere® MQ for Windows only.

In WebSphere MQ Version 6.0 all WebSphere MQ services were migrated to WebSphere MQ service or listener

objects, with the exception of ROOT custom services and queue manager services.

Use amqmdain to define and administer ROOT custom services, define, and administer queue manager services, and perform other Windows specific administrative tasks.

Starting a queue manager service with amqmdain is equivalent to using the strmqm command with the option `-ss`. amqmdain makes the queue manager run in a non-interactive session under a different user account. However, to ensure all queue manager startup feedback is returned to the command line, use the `strmqm -ss` command rather than amqmdain.

To administer and define WebSphere MQ service and listener objects, use MQSC commands, PCF commands, or the WebSphere MQ Explorer.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '2.1.0! -c \n' + '2.1.0?\n' + '2.1.0 -w \n' + '2.1.0 -i \n' + '2.1.0 -p \n' + '2.1.1? \n' + '2.1.1 -s \n' + '2.1.1 -r \n' + '2.1 -x \n' + '2 qmgr alter *QMgrName*\n' + '2.1? \n' + '2.1 -i *Initiation*\n' + '2.1 -x *Multi-instance*\n' + '2 refresh \n' + '2 svc list \n' + '2 svc view *ServiceName*\n' + '2 svc start *ServiceName*\n' + '2 svc end *ServiceName*\n' + '2 svc alter *ServiceName*\n' + '2.1 %Service options\n' + '2 svc delete *ServiceName*\n' + '2 svc define *ServiceName*\n' + '2.1 %Service options\n' + '2 auto *QMgrName*\n' + '2 manual *QMgrName*\n' + '2 status \n' + '2.1? \n' + '2.1 *QMgrName*\n' + '2.1 all\n' + '2 regsec \n' + '2 spn *QMgrName*\n' + '2.1 set\n' + '2.1 unset\n' + '2 reg \n' + '2.1? \n' + '2.1 *QMgrName*\n' + '2.1 * \n' + '2.2 *RegParams*\n' + '\n' + 'Service options\n' + '\n' + '1? -m *QMgrName*\n' + '2? -i *Initiation*\n' + '3? -t *Service*\n' + '4? -s command\n' + '5? -e command\n' + '6? -x *Execution*\n' + '\n' + '

');break; default:document.write('

');} //]]>

Keywords and parameters

All parameters are required unless the description states they are optional.

In every case, *QMgrName* is the name of the queue manager service to which the command applies, and *ServiceName* is the name of the ROOT custom services to which the command applies.

`qmgr start` *`QMgrName`*

Starts a queue manager service.

This parameter can also be written in the form *start QMgrName*.

If you start your queue manager as a service and need the queue manager to continue to run after logoff, use `strmqm -ss qmgr` instead of `amqmdain start qmgr`.

`qmgr end` *`QMgrName`*

Ends a queue manager service.

This parameter can also be written in the form *end QMgrName*.

For consistency across platforms, use `endmqm qmgr` instead of `amqmdain end qmgr`.

For fuller descriptions of the options, see endmqm.

`-c`

Controlled (or quiesced) shutdown.

`-w`

Wait shutdown.

`-i`

Immediate shutdown.

`-p`

Preemptive shutdown.

`-r`

Reconnect clients.

`-s`

Switch over to a standby queue manager instance.

`-x`

End the standby instance of the queue manager without ending the active instance.

`qmgr alter` *`QMgrName`*

Alters a queue manager service.

`-i` *`Initiation`*

Specifies the initiation type. Possible values are:

auto	Sets the queue manager service to automatic startup (when the machine starts up, or more precisely when the AMQMSRVN process starts up). The syntax is: `amqmdain qmgr alter QmgrName -i auto`
interactive	Sets the queue manager service to manual startup that then runs under the logged on (interactive) user. The syntax is: `amqmdain qmgr alter QmgrName -i interactive`
service	Sets the queue manager service to manual startup that then runs as a service. The syntax is: `amqmdain qmgr alter QmgrName -i service`

`-x` *`Multi-instance`*

Specifies if auto queue manager start by the WebSphere MQ service permits multiple instances. Equivalent to the `-sax` option on the crtmqm command. Also specifies if the amqmdain start `qmgr` command permits standby instances. Possible values are:

set	Sets automatic queue manager startup to permit multiple instances. Issues `strmqm` **`-x`**. The `set` option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is: `amqmdain qmgr alter QmgrName -x set`
unset	Sets automatic queue manager startup to single instance. Issues `strmqm`. The

`unset` option is ignored for queue managers that are initiated interactively or as a manual service startup. The syntax of the command is:

```
amqmdain qmgr alter QmgrName -x unset
```

refresh

Refreshes or checks the status of a queue manager. You will not see anything returned on the screen after executing this command.

svc list

Displays a list of currently defined ROOT custom services.

svc view *ServiceName*

Displays detailed information for a ROOT custom service.

svc start *ServiceName*

Starts a ROOT custom service.

svc end *ServiceName*

Ends a ROOT custom service.

svc delete *ServiceName*

Deletes a ROOT custom service.

svc alter *ServiceName ServiceOptions*

Alter a ROOT custom service with the options specified in *ServiceOptions*.

svc define *ServiceName ServiceOptions*

Define a ROOT custom service with the options specified in *ServiceOptions*.

auto *QMgrName*

Sets a queue manager service to automatic startup.

manual *QMgrName*

Sets a queue manager service to manual startup.

status *QMgrName* | all

These parameters are optional.

If no parameter is supplied:	Displays the status of the WebSphere MQ services.
If a *QMgrName* is supplied:	Displays the status of the named queue manager service.
If the parameter *all* is supplied:	Displays the status of all ROOT custom services.

regsec

Ensures that the security permissions assigned to the Registry keys are correct.

spn *QMgrName* set | unset

Allows you to set or unset the service principal name for a queue manager.

reg *QMgrName* | * *RegParams*

Parameters *QMgrName*, and * are optional.

If *RegParams* is specified alone:	Modifies **queue manager** configuration information in the Windows Registry related to the default queue manager.
If *QMgrName* and *RegParams* are specified:	Modifies **queue manager** configuration information in the Windows Registry related to the queue manager specified by *QMgrName*.
If * and *RegParams* are specified:	Modifies **WebSphere MQ** configuration information in the Windows Registry.

The parameter, *RegParams*, specifies the Registry stanzas to change, and the changes that are to be made. *RegParams* takes one of the following forms:

- **-c add -s *stanza* -v attribute=*value***
- **-c remove -s *stanza* -v [attribute|*]**
- **-c display -s *stanza* -v [attribute|*]**

If you are specifying queue manager configuration information, the valid values for *stanza* are:

XAResourceManager*name*
ApiExitLocal*name*
Channels
ExitPath

```
InstanceData
Log
QueueManagerStartup
TCP
LU62
SPX
NetBios
Connection
QMErrorLog
Broker
ExitPropertiesLocal
```

If you are modifying WebSphere MQ configuration information, the valid values for *stanza* are:

```
ApiExitCommon\name
ApiExitTemplate\name
ACPI
AllQueueManagers
Channels
DefaultQueueManager
LogDefaults
ExitProperties
```

The following are usage considerations:

- **amqmdain does not validate the values you specify for *name*, *attribute*, or *value*.**
- **When you specify `add`, and an attribute already exists, it is modified.**
- **If a stanza does not exist, amqmdain creates it.**
- **When you specify `remove`, you can use the value `*` to remove all attributes.**
- **When you specify `display`, you can use the value `*` to display all attributes which have been defined. Note that this only displays the attributes which have been defined and not the complete list of valid attributes.**
- **If you use `remove` to delete the only attribute in a stanza, the stanza itself is deleted.**
- **Any modification you make to the Registry re-secures all WebSphere MQ Registry entries.**

ServiceOptions

The options available when defining, or altering, a ROOT custom service.

-m *QMgrName*

The name of the associated queue manager.

If no parameter is supplied:	The service is defined as a ROOT custom service with no associated queue manager.
If *QMgrName* is supplied:	The queue manager specified by *QMgrName* is used.

If you omit this parameter, the service is defined as a ROOT custom service with no associated queue manager.

-i *Initiation*

Specifies the initiation type. Possible values are:

auto	Sets the ROOT custom service to automatic startup (when the machine starts up, or more precisely when the AMQMSRVN process starts up). For example: `amqmdain svc alter ServiceName -i auto`
interactive	Sets the ROOT custom service to manual startup that then runs under the logged on (interactive) user. For example: `amqmdain svc alter ServiceName -i interactive`
service	Sets the ROOT custom service to manual startup that then runs as a service. For example: `amqmdain svc alter ServiceName -i service`

`amqmdain auto QmgrName`

If you omit this parameter, automatic startup is set.

-t *Service*

Specifies the ROOT custom service type. Possible values are:

process	The ROOT custom service is not expected to run to completion. To end the ROOT custom service, issue the *svc end ServiceName* command.
command	The ROOT custom service is expected to run to completion.

If you omit this parameter, *Service* is specified as process.

-s *command*

The command to execute when the ROOT custom service starts.

-e *command*

The command to execute when the ROOT custom service ends.

-x *Execution*

Specifies the execution type. Possible values are:

prefix	The ROOT custom service starts before the associated queue manager starts.
suffix	The ROOT custom service starts after the associated queue manager starts.

If you omit this parameter, *Execution* is specified as suffix.

Examples

The following example adds an XAResourceManager to queue manager TEST. The commands issued are:

```
amqmdain reg TEST -c add -s XAResourceManager\Sample -v SwitchFile=sf1
amqmdain reg TEST -c add -s XAResourceManager\Sample -v ThreadOfControl=THREAD
amqmdain reg TEST -c add -s XAResourceManager\Sample -v XAOpenString=openit
amqmdain reg TEST -c add -s XAResourceManager\Sample -v XACloseString=closeit
```

To display the values set by the commands above, use:

```
amqmdain reg TEST -c display -s XAResourceManager\Sample -v *
```

The display would look something like this:

```
0784726, 5639-B43 (C) Copyright IBM Corp. 1994, 2002. ALL RIGHTS RESERVED.
Displaying registry value for Queue Manager 'TEST'
         Attribute = Name, Value = Sample
         Attribute = SwitchFile, Value = sf1
         Attribute = ThreadOfControl, Value = THREAD
         Attribute = XAOpenString, Value = openit
         Attribute = XACloseString, Value = closeit
```

To remove the XAResourceManager from queue manager TEST, use:

```
amqmdain reg TEST -c remove -s XAResourceManager\Sample -v *
```

Return codes

0	Command completed normally
-2	Syntax error
-3	Failed to initialize COM library
-4	Failed to initialize COM components
-7	Failed to configure service
-9	Unexpected Registry error
-10	Unable to access required service interface (IMQDService)
-11	Unable to access required service interface (ICustomService)
-12	Unable to access required service interface (ICustomServices)
-13	Unable to access required service interface (IUnknown)
-14	Specified service not found
-15	Specified service name already exists
-16	Failed to configure service principal name
-17	Failed to start service
-18	Failed to end service
-19	Failed to delete service
-20	Failed to store service definition
-21	Service initiation type could not be configured
-22	Service flags could not be configured
-23	Service flags could not be read
-24	Service dependency could not be configured
-25	Service start command could not be configured
-26	Service end command could not be configured
-27	Service name could not be configured

Note:

1. **If the *qmgr start QMgrName* command is issued, all return codes that can be returned with strmqm, can be**

returned here also. For a list of these return codes, see strmqm.

2. If the *qmgr end QMgrName* command is issued, all return codes that can be returned with endmqm, can be returned here also. For a list of these return codes, see endmqm.

Parent topic: The control commands

Related reference
Attributes for changing WebSphere MQ configuration information
Changing queue manager configuration information

This build: January 26, 2011 11:55:25

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15570_

5.2.4. amqmfsck (file system check)

Check that a file system is compliant with POSIX standards and capable of sharing queue manager data to support multi-instance queue managers.

Purpose

The amqmfsck command applies only to UNIX and IBM® i systems. You do not need to check the network drive on Windows.

If the check succeeds, the command reports `The tests on the directory completed successfully`.

If the check fails, the file system is not capable of being used by WebSphere® MQ queue managers. If the tests fail, choose verbose mode to assist you interpreting the errors.

The output from the `verbose` option helps you understand why the command failed, and if the problem can be solved by reconfiguring the file system.

Sometimes the failure might be an access control problem that can be fixed by changing directory ownership or permissions. Sometimes the failure can be fixed by reconfiguring the file system to behave in a different way. For example, some file systems have performance options that might need to be changed. It is also possible that the file system protocol does not support concurrency sufficiently robustly, and you need to use a different file system. For example, you need to use NFSv4 rather than NFSv3.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
+ '>>-amqmfsck--+-----+--+-----+--DirectoryName-------------------><\n'
+ '              \'- -v-\'   +- -w +                      \n'
+ '                         \'- -c \'                      \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

DirectoryName

The name of the directory to check.

Optional parameters

-c

Test writing to a file in the directory concurrently.

-v

Verbose output.

-w

Test waiting for and releasing locks.

Usage

The user must be a WebSphere MQ Administrator to run the command. You must have read/write access to the

directory being checked.

You need to run all three variants of the command to complete the testing,

1. **amqmfsck *DirectoryName***
2. **amqmfsck `-c` *DirectoryName***
3. **amqmfsck `-w` *DirectoryName***

When testing with the `-c` and `-w` options, run the same command on two machines at the same time.

On IBM i, use QSH to run the program. There is no CL command.

The command returns an exit code of zero, if the tests complete successfully.

Examples

The first set of three examples shows the command producing minimal output.

Successful test of basic file locking on one machine

```
> amqmfsck /shared/qmdata
The tests on the directory completed successfully.
```

Failed test of basic file locking on one machine

```
> amqmfsck /shared/qmdata
AMQ6245: Error Calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck' error '2'.
```

Successful test of locking on two machines

Table 1. Successful locking on two machines

Machine 1	Machine 2
> amqmfsck -w /shared/qmdata Please start this program on a second machine with the same parameters. Lock acquired. Press Return or terminate the program to release the lock.	
	> amqmfsck -w /shared/qmdata Waiting for lock...
[Return pressed] Lock released.	
	Lock acquired. The tests on the directory completed su

The second set of three examples shows the same commands using verbose mode.

Successful test of basic file locking on one machine

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")'
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd1 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd1, F_SETLK, F_RDLCK)
System call: fd2 = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd2, F_SETLK, F_RDLCK)
System call: close(fd2)
System call: write(fd1)
System call: close(fd1)
The tests on the directory completed successfully.
```

Failed test of basic file locking on one machine

```
> amqmfsck -v /shared/qmdata
System call: stat("/shared/qmdata")
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
```

```
System call: fchmod(fd, 0666)
System call: fstat(fd)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: write(fd)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_WRLCK)
System call: close(fd)
System call: fd = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fd, F_SETLK, F_RDLCK)
System call: fdSameFile = open("/shared/qmdata/amqmfsck.lck", O_RDWR, 0666)
System call: fcntl(fdSameFile, F_SETLK, F_RDLCK)
System call: close(fdSameFile)
System call: write(fd)
AMQxxxx: Error calling 'write()[2]' on file '/shared/qmdata/amqmfsck.lck', errno 2
(Permission denied).
```

Successful test of locking on two machines

Table 2. Successful locking on two machines - verbose mode

Machine 1	Machine 2

`> amqmfsck -wv /shared/qmdata`
`Calling 'stat("/shared/qmdata")'`
`Calling 'fd = open("/shared/qmdata/amqmfsck.lkw",`
`O_EXCL | O_CREAT | O_RDWR, 0666)'`
`Calling 'fchmod(fd, 0666)'`
`Calling 'fstat(fd)'`
`Please start this program on a second`
`machine with the same parameters.`
`Calling 'fcntl(fd, F_SETLK, F_WRLCK)'`
`Lock acquired.`
`Press Return`
`or terminate the program to release the lock.`	

	`> amqmfsck -wv /shared/qmdata`
	`Calling 'stat("/shared/qmdata")'`
	`Calling 'fd = open("/shared/qmdata/a`
	`O_EXCL | O_CREAT | O_RDWR,0666)'`
	`Calling 'fd = open("/shared/qmdata/a`
	`O_RDWR, 0666)'`
	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`'Waiting for lock...`

`[ Return pressed ]`
`Calling 'close(fd)'`
`Lock released.`	

	`Calling 'fcntl(fd, F_SETLK, F_WRLCK)`
	`Lock acquired.`
	`The tests on the directory completed`

Parent topic: The control commands

This build: January 26, 2011 11:55:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15575_

5.2.5. amqtcert

Migrate certificates from WebSphere MQ 5.3 or 5.3.1 (Windows only).

Purpose

The amqtcert command applies to WebSphere® MQ for Windows only.

The amqtcert command is used to migrate SSL Certificates from WebSphere MQ for Windows Version 5.3, or Version 5.3.1. SSL Certificate Migration instructions are detailed in the WebSphere MQ Migration Information. SSL Certificate Migration occurs after migrating WebSphere MQ for Windows Version 5.3, or Version 5.3.1.

In this section when referring to a WebSphere MQ Certificate Store file, we are specifically referring to a WebSphere MQ for Windows Version 5.3, or Version 5.3.1, Certificate Store file.

To use this command, you must be either an administrator or a member of the mqm group.

The amqtcert command is used to migrate certificates from a client's or queue manager's WebSphere MQ Certificate Store file to a GSKit key database file. The file name of the WebSphere MQ Certificate Store file is of the form *xxx*.sto, where *xxx* is your chosen name. The file name of the GSKit key database file is of the form *yyy*.kdb, where *yyy* is your chosen name.

The amqtcert command is used to perform the following types of migration:

Automatic migration

The migration is deferred.

The time at which the migration occurs depends on whether it is being done for a queue manager or a WebSphere MQ client. On a queue manager the migration occurs when the queue manager starts. On a WebSphere MQ client the migration occurs when the first SSL channel starts.

Manual migration

The migration occurs immediately.

The command is also used to set the state information relating to automatic migration, held in the Windows registry, for each queue manager or client.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

ExpTime\n' + '2.2 –c *FileName*\n' + '2.2 –m *QMgrName*\n' + '2.2 –m *\n' + '2 –g *FileName* –w *FileName*\n' + '2.1 %Manual migration options\n' + '2 –l \n' + '2.1 –a \n' + '2.1 –c *FileName*\n' + '2.1 –m *QMgrName*\n' + '2 –r \n' + '2.1 –c *\n' + '2.1 –c *FileName*\n' + '2.1 –m *QMgrName*\n' + '2.1 –m *\n' + '\n' + 'Manual migration options\n' + '\n' + '1 –p *Password*\n' + '2? –e *ExpTime*\n' + '3? \n' + '3 –u *ClntLogonID* –i *ListNumber*\n' + '3 –m *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-amqtcert-------------------------------------------------->\n'
+ '\n'
+ '>--+- -a -- -p --Password--+---------------+--+- -c --FileName-+--+-><\n'
+ '    |                          \'- -e --ExpTime-\'   +- -m --QMgrName-+   |    \n'
+ '    |                                                \'- -m *----------\'   |    \n'
+ '    +- -g --FileName-- -w --FileName--| Manual migration options |-+    \n'
+ '    +- -l --+- -a -----------+--------------------------------------+    \n'
+ '    |        +- -c --FileName-+                                           |    \n'
+ '    |        \'- -m --QMgrName-\'                                           |    \n'
+ '    \'- -r --+- -c *----------+--------------------------------------\'    \n'
+ '             +- -c --FileName-+                                              \n'
+ '             +- -m --QMgrName-+                                              \n'
+ '             \'- -m *----------\'                                                \n'
+ '\n'
+ 'Manual migration options\n'
+ '\n'
+ '|-- -p --Password--+---------------+--------------------------->\n'
+ '                    \'- -e --ExpTime-\'    \n'
+ '\n'
+ '>--+-------------------------------------+---------------------|\n'
+ '    +- -u --ClntLogonID-- -i --ListNumber-+    \n'
+ '    \'- -m --QMgrName----------------------\'    \n'
+ '\n'
+ '
```

');} //]]>

Keywords and parameters

-a

Specifies automatic migration.

When used in conjunction with the -m or -c parameters, it prepares the specified queue manager or client to automatically migrate the WebSphere MQ Certificate Store.

When used in conjunction with the -l parameter, it lists the contents of the registry entries for automatic migration.

-c *FileName*|*

» *FileName* specifies the absolute (rather than relative) directory path name and file name (excluding the .sto suffix) of the client's WebSphere MQ Certificate Store. If there are any spaces in *FileName* then it must be enclosed in quotes. In manual migration, the -c parameter is not required.

» *FileName* is used to identify a specific client WebSphere MQ Certificate Store. For automatic migration, the file name is stored in the registry and flagged as requiring automatic migration. «

» When the client connects to the queue manager, the key repository value (either MQSSLKEYR or the KeyRepository field of the MQSCO) being used by the client is compared against the list of stored file names flagged as requiring automatic migration; if the values match then migration takes place. The file name is cleared from the registry list once successful migration has taken place. «

-c * is used only in combination with the -r flag and specifies all client entries in the registry.

«

-e *ExpTime*

The expiration time (in days) of the GSKit key database password. The default is 60 days.

-g *Filename*

» Use manual migration. The absolute (rather than relative) directory path name and file name (excluding the .kdb suffix) of a GSKit key database. If there are any spaces in *FileName* then it must be enclosed in quotes. The -w parameter must also be specified. «

-l

In combination with the -c *FileName* or -m *QMgrName* parameters, it lists the certificates in a WebSphere MQ Certificate Store.

In combination with the -a parameter, it lists the contents of the registry entries for automatic migration.

-m *QMgrName*|*

QMgrName specifies the name of an individual queue manager. * represents all queue managers.

When specifying manual migration of a queue manager certificate store, the -m *QMgrName* parameter is mandatory. This allows the correct label to be given to the assigned personal certificate when it is written to the GSkit key database file (see the description of the -u parameter for more details). The * value is not valid for manual migration.

When specifying automatic migration, the names of the source certificate store and the target key database file are derived from the queue manager's SSLKeyRepository attribute.

-p *Password*

The password for the GSKit key database. This must be specified for automatic or manual migration. The maximum password length is 255 bytes.

-r

Remove the registry state information relating to automatic migration.

-u *ClntLogonID*

This parameter is only applicable when the command is used for manual migration of clients. The *-i ListNumber* parameter must also be specified.

In the WebSphere MQ Certificate Store there is usually one certificate assigned to the client. During migration, the copy of this certificate is modified before it is stored in the GSKit database.

The modification sets the certificate's Friendly Name attribute to the string `ibmwebspheremq`, followed in lower case by the client logon ID. The previous Friendly Name value, if any, is lost. This Friendly Name value becomes the label in the GSKit key database.

If neither -u nor -m are specified on manual migration, it is assumed to be a client migration. The *ClntLogonId* used is the userid used by the current amqtcert user to logon.

-i *ListNumber*

This parameter is only applicable when the command is used for manual migration of clients. The *-u ClntLogonID* parameter must also be specified.

This parameter is used to identify a specific personal certificate which is to have its GSKit label set to the value specified by the *-u ClntLogonID* parameter.

Prior to using amqtcert with *-i ListNumber* specified, you must execute amqtcert with *-l* specified to list the certificates in a WebSphere MQ Certificate Store. You must identify the required personal certificate from the list, then execute amqtcert again, specifying *-i ListNumber* with the required certificate number.

For example, after executing `amqtcert -l -c C:\SSL\Client\key` you might identify the following personal certificate from the list displayed as the required certificate:

```
Certificate 14
Certificate Type:  Personal
```

```
Subject:            personalcert@ibm.com, personalcert@ibm.com
Issuer:             BE, GlobalSign nv-sa, PersonalSign Class 1 CA, GlobalSign
                                                          PersonalSign Class 1 CA
Valid From:         14/10/2004 to 14/11/2004
Certificate Usage: <All>
```

You will then execute amqtcert and specify *-i ListNumber* as `-i 14`.

***ListNumber* must be a number greater than 0.**

If *ListNumber* references a valid personal certificate, which is not the currently assigned certificate, then:

- **The assigned certificate is not modified.**
- **The assigned certificate is not given a label of the form `ibmwebspheremq<xxxxx>` in the GSkit key database file, and ceases to be assigned.**
- **The certificate referenced by *ListNumber* becomes the assigned certificate in the GSKit key database.**

If *ListNumber* does not reference a valid personal certificate, then the command fails and no migration occurs for any certificates (personal or otherwise).

-w *FileName*

Use manual migration. » *FileName* « is the absolute (rather than relative) directory path name and file name (excluding the .sto suffix) of a WebSphere MQ Certificate Store. If there are any spaces in *FileName* then it must be enclosed in quotes. The -g parameter must also be specified.

Examples

Return codes

1	Error accessing certificate store
2	Auto migration failed
3	Invalid argument combination
4	Certificate expired
5	Certificate import failed
6	Certificate is an orphan
7	Create file failed
8	Duplicate registry entry
9	WebSphere MQ Certificate Store file is empty
16	WebSphere MQ Certificate Store file found
17	WebSphere MQ Certificate Store file not found
18	GSKit add certificate failed
19	GSKit error
20	GSKit initialization error
21	GSkit add CA certificate error
22	Load library failed
23	No memory to allocate tables for migrating root/intermediate certificates
24	No memory
25	WebSphere MQ Certificate Store file cannot be opened
32	User not authorized to run **amqtcert** command
33	Windows operation failed
34	Windows export of personal certificate failed
35	GSKit create new key database error
36	Windows registry error
37	**amqtcert** command usage error
38	Queue manager name error
39	Unexpected system return code
40	Local mqm group not found
41	Invalid arguments
48	Bad argument
49	Invalid *-i ListNumber* parameter

Related commands

amqccert	Check certificate chains

Listing the contents of certificate stores

Manually migrating certificate stores

Automatically migrating certificate stores

Listing the contents of registry entries

Removing state information

Parent topic: The control commands

This build: January 26, 2011 11:55:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15580_

5.2.5.1. Listing the contents of certificate stores

```
amqtcert -l -c C:\SSL\Client\key
```

Lists the contents of the client's WebSphere® MQ Certificate Store.

```
amqtcert -l -m QM1
```

Lists the contents of the QM1 queue manager's WebSphere MQ Certificate Store.

Parent topic: amqtcert

This build: January 26, 2011 11:55:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15590_

5.2.5.2. Manually migrating certificate stores

```
amqtcert -g C:\SSL\Client\key -w C:\SSL\Client\key -p MyPassword
```

Manually migrates the client WebSphere® MQ Certificate Store, specified by the -w parameter value, to the GSKit key database specified by the -g parameter value. It sets the password of the GSKit key database (called key.kdb) to `MyPassword`. It also sets the GSkit label (of the certificate which was assigned in the WebSphere MQ client store) to `ibmwebspheremq<mylogonid>`, where *<mylogonid>* stands for the logon id of the current user in lower case.

```
amqtcert -g C:\GSKitSSL\Client\key -w C:\SSL\Client\key -p MyPassword -u MyClientID -i 14
```

Manually migrates a client's WebSphere MQ Certificate Store, specified by the -w parameter value, to the GSKit key database specified by the -g parameter value. This command sets the password of the GSKit key database (named key.kdb) to `MyPassword`. It also sets the GSKit label of the certificate specified by -i to `ibmwebspheremq<mylogonid>`, where *<mylogonid>* stands for the logon id of the current user in lower case.

```
amqtcert -g "C:\Program Files\IBM\WebSphere MQ\Qmgrs\QM1\SSL\key" -w "C:\Program
Files\IBM\WebSphere MQ\Qmgrs\QM1\SSL\key" -p MyPassword -m QM1
```

Manually migrates the WebSphere MQ Certificate Store for queue manager QM1, specified by the -w parameter value, to the GSKit key database specified by the -g parameter value. It sets the password of the GSKit key database (named key.kdb) to `MyPassword`. It also sets the GSkit label (of the certificate which was assigned to queue manager QM1) to `ibmwebspheremqqm1`.

Parent topic: amqtcert

This build: January 26, 2011 11:55:26

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15600_

5.2.5.3. Automatically migrating certificate stores

```
amqtcert -a -p MyPassword -m QM1
```

Automatically migrates the WebSphere® MQ Certificate Store for queue manager QM1, and sets the GSKit key database password to `MyPassword`

```
amqtcert -a -p MyPassword -c C:\SSL\Client\key
```

Automatically migrates the specified client WebSphere MQ Certificate Store, and sets the GSKit key database password to "MyPassword".

```
amqtcert -a -p MyPassword -m *
```

Automatically migrates the WebSphere MQ Certificate Stores for all queue managers, and sets the GSKit key database password to `MyPassword`.

Parent topic: amqtcert

This build: January 26, 2011 11:55:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15610_

5.2.5.4. Listing the contents of registry entries

```
amqtcert -l -a
```

Lists the contents of the registry entries for automatic migration.

Parent topic: amqtcert

This build: January 26, 2011 11:55:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15620_

5.2.5.5. Removing state information

```
amqtcert -r -c C:\SSL\Client\key
```

Removes the registry state information relating to automatic migration for the specified client WebSphere® MQ Certificate Store.

```
amqtcert -r -c *
```

Removes the registry state information relating to automatic migration for all clients.

```
amqtcert -r -m QM1
```

Removes the registry state information relating to automatic migration for queue manager QM1.

```
amqtcert -r -m *
```

Removes the registry state information relating to automatic migration for all queue managers.

Parent topic: amqtcert

This build: January 26, 2011 11:55:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15630_

5.2.6. crtmqcvx

Create data conversion code from data type structures.

Purpose

Use the crtmqcvx command to create a fragment of code that performs data conversion on data type structures. The command generates a C function that can be used in an exit to convert C structures.

The command reads an input file containing structures to be converted, and writes an output file containing code fragments to convert those structures.

For information about using this command, see the WebSphere MQ Application Programming Guide.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
TargetFile\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-crtmqcvx--SourceFile--TargetFile----------------------------><\n'
+ '\n'
+ '
```

');} //]]>

Required parameters

SourceFile

The input file containing the C structures to convert.

TargetFile

The output file containing the code fragments generated to convert the structures.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

Examples

The following example shows the results of using the data conversion command against a source C structure. The command issued is:

```
crtmqcvx source.tmp target.c
```

The input file, `source.tmp` looks like this:

```
/* This is a test C structure which can be converted by the */
/* crtmqcvx utility                                         */

struct my_structure
{
     int     code;
     MQLONG value;
};
```

The output file, `target.c`, produced by the command is shown below. You can use these code fragments in your applications to convert data structures. However, if you do so, the fragment uses macros supplied in the header file `amqsvmha.h`.

```
MQLONG Convertmy_structure(
               PMQBYTE *in_cursor,
               PMQBYTE *out_cursor,
               PMQBYTE in_lastbyte,
               PMQBYTE out_lastbyte,
               MQHCONN hConn,
               MQLONG  opts,
               MQLONG  MsgEncoding,
               MQLONG  ReqEncoding,
               MQLONG  MsgCCSID,
               MQLONG  ReqCCSID,
               MQLONG  CompCode,
               MQLONG  Reason)
{
     MQLONG ReturnCode = MQRC_NONE;

     ConvertLong(1); /* code */

     AlignLong();
     ConvertLong(1); /* value */

Fail:
     return(ReturnCode);
}
```

Parent topic: The control commands

This build: January 26, 2011 11:55:27

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15640_

5.2.7. crtmqm

»Create a queue manager.«

Purpose

»Use the crtmqm command to create a queue manager and define the default and system objects. The objects created by crtmqm are listed in System and default objects. When a queue manager has been created, use the strmqm command to start it.«

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

DefaultTransmissionQueue\n' + '4? -h *MaximumHandleLimit*\n' + '5! -lc \n' + '5?\n' + '5 -ll \n' + '6? -ld *LogPath*\n' + '7? -lf *LogFilePages*\n' + '8? -lp *LogPrimaryFiles*\n' + '9? -ls *LogSecondaryFiles*\n' + '10? -q \n' + '11! -ss \n' + '11?\n' + '11 -sa \n' + '11 -sax \n' + '11 -si \n' + '12? -md *DataPath*\n' + '13? -g *ApplicationGroup*\n' + '14? -t *IntervalValue*\n' + '15? -u *DeadLetterQueue*\n' + '16? -x *MaximumUncommittedMessages*\n' + '17? -z \n' + '18 *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-crtmqm--+------------+--+--------------------------------+--->\n'
+ '            \'- -c --Text-\'   \'- -d --DefaultTransmissionQueue-\'    \n'
+ '\n'
+ '                                     .- -lc -.    \n'
+ '>--+---------------------------+--+-------+----------------------->\n'
+ '    \'- -h --MaximumHandleLimit-\'   \'- -ll -\'    \n'
+ '\n'
+ '>--+-----------------+--+---------------------+------------------->\n'
+ '    \'- -ld --LogPath-\'   \'- -lf --LogFilePages-\'    \n'
+ '\n'
+ '>--+-------------------------+--+--------------------------+-----><\n'
+ '    \'- -lp --LogPrimaryFiles-\'   \'- -ls --LogSecondaryFiles-\'    \n'
+ '\n'
+ '                .- -ss --.                              \n'
+ '>--+------+--+--------+--+-----------------+----------------------->\n'
+ '    \'- -q -\'   +- -sa --+   \'- -md -DataPath-\'    \n'
+ '                +- -sax -+                              \n'
+ '                \'- -si --\'                              \n'
+ '\n'
+ '>--+------------------------+--+--------------------+---------->\n'
+ '    \'- -g --ApplicationGroup-\'   \'- -t --IntervalValue-\'    \n'
+ '\n'
+ '>--+------------------------+-----------------------------------><\n'
+ '    \'- -u --DeadLetterQueue-\'    \n'
+ '\n'
+ '>--+-----------------------------------+--+------+--QMgrName---><\n'
+ '    \'- -x --MaximumUncommittedMessages-\'   \'- -z -\'                \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

QMgrName

The name of the queue manager to create. The name can contain up to 48 characters. This parameter must be the last item in the command.

Optional parameters

-c *Text*

Descriptive text for this queue manager. You can use up to 64 characters; the default is all blanks.

If you include special characters, enclose the description in double quotes. The maximum number of characters is

reduced if the system is using a double-byte character set (DBCS).

-d *DefaultTransmissionQueue*

The name of the local transmission queue where remote messages are put if a transmission queue is not explicitly defined for their destination. There is no default.

-g *ApplicationGroup*

Enable Restricted mode operation for this queue manager. ApplicationGroup is the name of the group containing members allowed to use the queue manager.
This option applies only to WebSphere® MQ for AIX®, Solaris, HP-UX, and Linux. The default value is `-g all` which allows unrestricted access.
The **-g** *ApplicationGroup* value is recorded in the queue manager configuration file, `qm.ini`.
The mqm user ID and the user executing the command must belong to the specified ApplicationGroup.
See Restricted mode for more information.

-h *MaximumHandleLimit*

The maximum number of handles that any one application can have open at the same time.

Specify a value in the range 1 through 999999999. The default value is 256.

The next six parameter descriptions relate to logging, which is described in Using the log for recovery.

Note: Choose the logging arrangements with care, because some cannot be changed once they are committed.

-lc

Use circular logging. Circular logging is the default logging method.

-ld *LogPath*

The directory used to hold log files.

In WebSphere MQ for Windows, the default is `C:\Program Files\IBM\WebSphere MQ\log` (assuming that C is your data drive).

In WebSphere MQ for UNIX systems, the default is `/var/mqm/log`.

User ID mqm and group mqm must have full authorities to the log files. If you change the locations of these files, you must give these authorities yourself. This occurs automatically if the log files are in their default locations.

-lf *LogFilePages*

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

In WebSphere MQ for UNIX systems, the default number of log file pages is 4096, giving a log file size of 16 MB. The minimum number of log file pages is 64 and the maximum is 65535.

In WebSphere MQ for Windows systems, the default number of log file pages is 4096, giving a log file size of 16 MB. The minimum number of log file pages is 32 and the maximum is 65535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

-ll

Use linear logging.

-lp *LogPrimaryFiles*

The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

Operating system limits can reduce the maximum log size.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

For more information about primary log files, see What logs look like.

To calculate the size of the primary log files, see Calculating the size of the log.

-ls *LogSecondaryFiles*

The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX systems. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

Operating system limits can reduce the maximum log size.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

For more information about the use of secondary log files, see What logs look like.

To calculate the size of the secondary log files, see Calculating the size of the log.

-md *DataPath*

The directory used to hold the data files for a queue manager.

In WebSphere MQ for Windows, the default is `C:\Program Files\IBM\WebSphere MQ\qmgrs` (assuming that C is your data drive). If the volume supports file security, the SYSTEM ID, Administrators, and mqm group must be granted read/write access to the directory.

In WebSphere MQ for UNIX systems, the default is `/var/mqm/qmgrs`.

The `DataPath` parameter is provided to assist in the configuration of multi-instance queue managers. For example on Unix systems; if `/var/mqm` is located on a local file system, use `DataPath` with `LogPath` to point to the shared file systems accessible to multiple queue managers.

Note: A queue manager created using the `DataPath` parameter does run on WebSphere MQ versions earlier than 7.0.1, but it needs reconfiguring to remove the `DataPath` parameter from its configuration. You have two options to restore the queue manager to a pre-version 7.0.1 configuration and run without the `DataPath` parameter. If you are confident about editing queue manager configurations, you can manually configure the queue manager using the `Prefix` queue manager configuration parameter. Alternatively,

1. Stop the queue manager.
2. Save the queue manager data and log directories.
3. Delete the queue manager.
4. Backout WebSphere MQ to the pre-v7.0.1 fix level.
5. Create the queue manager with the same name.
6. Replace the new queue manager data and log directories with the ones you saved.

-q

Makes this queue manager the default queue manager. The new queue manager replaces any existing default queue manager.

If you accidentally use this flag and want to revert to an existing queue manager as the default queue manager, change the default queue manager as described in Making an existing queue manager the default.

-sa

Automatic queue manager startup. For Windows systems only.

The queue manager is configured to start automatically when the machine starts up, or more precisely, when the AMQMSRVN process starts up.

This is the default option if you create a queue manager from WebSphere MQ Explorer.

Queue managers created in WebSphere MQ releases earlier than V7 retain their existing startup type.

-sax

Automatic queue manager startup, permitting multiple instances. For Windows systems only.

The queue manager is configured to start automatically when the machine starts up, or more precisely, when the AMQMSRVN process starts up.

If an instance of the queue manager is not already running elsewhere, the queue manager starts, the instance becomes active, and standby instances are permitted elsewhere. If a queue manager instance that permits standbys is already active on a different server, the new instance becomes a standby.

Only one instance of a queue manager runs on a server.

Queue managers created in WebSphere MQ releases earlier than V7.0.1 retain their existing startup type.

-si

Interactive (manual) queue manager start up.

The queue manager is configured to start only when manually requested using the strmqm command. The queue manager then runs under the logged on (interactive) user. Queue managers configured with interactive startup end when the user who started them logs off.

-ss

Service (manual) queue manager startup.

A queue manager configured to start only when manually requested using the strmqm command. The queue manager then runs as a child of the AMQMSRVN process. Queue managers configured with service startup continue to run even after the interactive user has logged off.

This is the default option if you create a queue manager from the command line.

-t *IntervalValue*

The trigger time interval in milliseconds for all queues controlled by this queue manager. This value specifies the time after receiving a trigger-generating message when triggering is suspended. That is, if the arrival of a message on a queue causes a trigger message to be put on the initiation queue, any message arriving on the same queue within the specified interval does not generate another trigger message.

You can use the trigger time interval to ensure that your application is allowed sufficient time to deal with a trigger condition before it is alerted to deal with another on the same queue. You might choose to see all trigger events that happen; if so, set a low or zero value in this field.

Specify a value in the range 0 through 999999999. The default is 999999999 milliseconds, a time of more than 11 days. Allowing the default to be used effectively means that triggering is disabled after the first trigger message. However, an application can enable triggering again by servicing the queue using a command to alter the queue to reset the trigger attribute.

-u *DeadLetterQueue*

The name of the local queue that is to be used as the dead-letter (undelivered-message) queue. Messages are put on this queue if they cannot be routed to their correct destination.

The default is no dead-letter queue.

-x *MaximumUncommittedMessages*

The maximum number of uncommitted messages under any one sync point. That is, the sum of:

- **The number of messages that can be retrieved from queues**
- **The number of messages that can be put on queues**
- **Any trigger messages generated within this unit of work**

This limit does not apply to messages that are retrieved or put outside a sync point.

Specify a value in the range 1 through 999999999. The default value is 10000 uncommitted messages.

-z

Suppresses error messages.

This flag is used within WebSphere MQ to suppress unwanted error messages. Because using this flag can result in loss of information, do not use it when entering commands on a command line.

Return codes

0	Queue manager created
8	Queue manager exists
39	Invalid parameter specified
49	Queue manager stopping
69	Storage not available
70	Queue space not available
71	Unexpected error
72	Queue manager name error
74	The WebSphere MQ service is not started.
100	Log location invalid
111	Queue manager created. However, there was a problem processing the default queue manager definition in the product configuration file. The default queue manager specification might be incorrect.
115	Invalid log size
119	Permission denied (Windows only)

Examples

1. **This command creates a default queue manager called `Paint.queue.manager`, with a description of `Paint`**

shop, and creates the system and default objects. It also specifies that linear logging is to be used:

```
crtmqm -c "Paint shop" -ll -q Paint.queue.manager
```

2. This command creates a default queue manager called Paint.queue.manager, creates the system and default objects, and requests two primary and three secondary log files:

```
crtmqm -c "Paint shop" -ll -lp 2 -ls 3 -q Paint.queue.manager
```

3. This command creates a queue manager called travel, creates the system and default objects, sets the trigger interval to 5000 milliseconds (or 5 seconds), and specifies SYSTEM.DEAD.LETTER.QUEUE as its dead-letter queue.

```
crtmqm -t 5000 -u SYSTEM.DEAD.LETTER.QUEUE travel
```

4. »This command creates a queue manager called QM1 on a UNIX or Linux system with its log and queue manager data folders in a common parent directory. The parent directory is shared on highly available networked storage to create a multi-instance queue manager. Prior to issuing the command, create /MQHA, /MQHA/logs and /MQHA/qmgrs owned by user and group mqm, and with permissions rwxrwxr-x.

```
crtmqm -ld /MQHA/logs -md /MQHA/qmgrs QM1
```

«

Related commands

strmqm	Start queue manager
endmqm	End queue manager
dltmqm	Delete queue manager

Parent topic: The control commands

This build: January 26, 2011 11:55:27

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15650_

5.2.8. dltmqm

»Delete a queue manager.«

Purpose

Use the dltmqm command to delete a specified queue manager and all objects associated with it. Before you can delete a queue manager you must end it using the endmqm command.

In WebSphere® MQ for Windows, it is an error to delete a queue manager when queue manager files are open. If you get this error, close the files and reissue the command.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
+ '>>-dltmqm--+------+--QMgrName-----------------------------------><\n'
+ '            \'- -z -\'               \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

QMgrName

The name of the queue manager to delete.

Optional parameters

-z

Suppresses error messages.

Return codes

0	Queue manager deleted
3	Queue manager being created
5	Queue manager running
16	Queue manager does not exist
24	A process that was using the previous instance of the queue manager has not yet disconnected.
25	An error occurred while creating or checking the directory structure for the queue manager.
26	Queue manager running as a standby instance.
27	Queue manager could not obtain data lock.
28	Queue manager deleted, however there was a problem removing it from the Service Control Manager.
29	Queue manager deleted, however there was a problem removing it from Active Directory.
33	An error occurred while deleting the directory structure for the queue manager.
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
74	The WebSphere MQ service is not started.
100	Log location invalid.
112	Queue manager deleted. However, there was a problem processing the default queue manager definition in the product configuration file. The default queue manager specification might be incorrect.
119	Permission denied (Windows only).

Examples

1. **The following command deletes the queue manager `saturn.queue.manager`.**

   ```
   dltmqm saturn.queue.manager
   ```

2. **The following command deletes the queue manager `travel` and also suppresses any messages caused by the command.**

   ```
   dltmqm -z travel
   ```

Usage notes

In WebSphere MQ for Windows, it is an error to delete a queue manager when queue manager files are open. If you get this error, close the files and reissue the command.

Deleting a cluster queue manager does not remove it from the cluster. To check whether the queue manager you want to delete is part of a cluster, issue the command DIS CLUSQMGR(*) and check if this queue manager is listed in the output. If it is listed as a cluster queue manager you must remove the queue manager from the cluster before deleting it. See the related link for instructions.

If you do delete a cluster queue manager without first removing it from the cluster, the cluster will continue to regard the deleted queue manager as a member of the cluster for at least 30 days. You can remove it from the cluster using the command RESET CLUSTER on a full repository queue manager. Note that recreating a queue manager with an identical name and then trying to remove that queue manager from the cluster does not result in the cluster queue manager being removed from the cluster, because the newly created queue manager, although having the same name, does not have the same queue manager ID (QMID) and is therefore treated as a different queue manager by the cluster.

Related commands

crtmqm	Create queue manager
strmqm	Start queue manager
endmqm	End queue manager

Parent topic: The control commands

Related information
Task 10: Removing a queue manager from a cluster

This build: January 26, 2011 11:55:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15660_

5.2.9. dmpmqaut

»Dump a list of current authorizations for a range of WebSphere MQ object types and profiles.«

Purpose

Use the dmpmqaut command to dump the current authorizations to a specified object.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

Profile\n' + '3 -l \n' + '4? -t *ObjectType*\n' + '5? -s *ServiceComponent*\n' + '6? \n' + '6 -p *PrincipalName*\n' + '6 -g *GroupName*\n' + '7? \n' + '7 -e \n' + '7 -x \n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dmpmqaut--+----------------+--+---------------+-------------->\n'
+ '              \'- -m --QMgrName-\'  +- -n --Profile-+    \n'
+ '                                    \'- -l ----------\'    \n'
+ '\n'
+ '>--+-------------------+--+-----------------------+------------->\n'
+ '   \'- -t --ObjectType-\'  \'- -s --ServiceComponent-\'    \n'
+ '\n'
+ '>--+---------------------+--+------+--------------------------><\n'
+ '   +- -p --PrincipalName-+  +- -e -+    \n'
+ '   \'- -g --GroupName-----\'  \'- -x -\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-m *QMgrName*

Dump authority records only for the queue manager specified. If you omit this parameter, only authority records for the default queue manager are dumped.

-n *Profile*

The name of the profile for which to dump authorizations. The profile name can be generic, using wildcard characters to specify a range of names as explained in Using OAM generic profiles on UNIX systems and Windows.

-l

Dump only the profile name and type. Use this option to generate a *terse* list of all defined profile names and types.

-t *ObjectType*

The type of object for which to dump authorizations. Possible values are:

authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
listener or **lstr**	A listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter
qmgr	A queue manager
service or **srvc**	A service
topic or **top**	A topic

-s *ServiceComponent*

If installable authorization services are supported, specifies the name of the authorization service for which to dump authorizations. This parameter is optional; if you omit it, the authorization inquiry is made to the first installable component for the service.

-p *PrincipalName*

This parameter applies to WebSphere® MQ for Windows only; UNIX systems keep only group authority records.

The name of a user for whom to dump authorizations to the specified object. The name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see Principals and groups.

-g *GroupName*

The name of the user group for which to dump authorizations. You can specify only one name, which must be the name of an existing user group. On Windows systems, you can use only local groups.

-e

Display all profiles used to calculate the cumulative authority that the entity has to the object specified in `-n Profile`. The variable `Profile` must not contain any wildcard characters.

The following parameters must also be specified:

- `-m QMgrName`
- `-n Profile`
- `-t ObjectType`

and either `-p PrincipalName`, or `-g GroupName`.

-x

Display all profiles with exactly the same name as specified in `-n Profile`. This option does not apply to the QMGR object, so a dump request of the form `dmpmqaut -m QM -t QMGR ... -x` is not valid.

Examples

The following examples show the use of dmpmqaut to dump authority records for generic profiles:

1. This example dumps all authority records with a profile that matches queue a.b.c for principal user1.

   ```
   dmpmqaut -m qm1 -n a.b.c -t q -p user1
   ```

   The resulting dump would look something like this:

   ```
   profile:      a.b.*
   object type:  queue
   entity:       user1
   type:         principal
   authority:    get, browse, put, inq
   ```

   Note: UNIX users cannot use the `-p` option; they must use `-g groupname` instead.

2. This example dumps all authority records with a profile that matches queue a.b.c.

   ```
   dmpmqaut -m qmgr1 -n a.b.c -t q
   ```

   The resulting dump would look something like this:

   ```
   profile:      a.b.c
   object type:  queue
   entity:       Administrator
   type:         principal
   authority:    all
   - - - - - - - - - - - - - - - - - -
   profile:      a.b.*
   object type:  queue
   entity:       user1
   type:         principal
   authority:    get, browse, put, inq
   - - - - - - - - - - - - - - - - - -
   profile:      a.**
   object type:  queue
   entity:       group1
   type:         group
   authority:    get
   ```

3. This example dumps all authority records for profile a.b.*, of type queue.

   ```
   dmpmqaut -m qmgr1 -n a.b.* -t q
   ```

   The resulting dump would look something like this:

   ```
   profile:      a.b.*
   object type:  queue
   entity:       user1
   type:         principal
   authority:    get, browse, put, inq
   ```

4. This example dumps all authority records for queue manager qmX.

```
dmpmqaut -m qmX
```

The resulting dump would look something like this:

```
profile:      q1
object type:  queue
entity:       Administrator
type:         principal
authority:    all
- - - - - - - - - - - - - - - -
profile:      q*
object type:  queue
entity:       user1
type:         principal
authority:    get, browse
- - - - - - - - - - - - - - - -
profile:      name.*
object type:  namelist
entity:       user2
type:         principal
authority:    get
- - - - - - - - - - - - - - - -
profile:      pr1
object type:  process
entity:       group1
type:         group
authority:    get
```

5. This example dumps all profile names and object types for queue manager qmX.

```
dmpmqaut -m qmX -l
```

The resulting dump would look something like this:

```
profile: q1, type: queue
profile: q*, type: queue
profile: name.*, type: namelist
profile: pr1, type: process
```

Note:

1. For WebSphere MQ for Windows only, all principals displayed include domain information, for example:

```
profile:      a.b.*
object type:  queue
entity:       user1@domain1
type:         principal
authority:    get, browse, put, inq
```

2. Each class of object has authority records for each group or principal. These records have the profile name `@CLASS` and track the `crt` (create) authority common to all objects of that class. If the `crt` authority for any object of that class is changed then this record is updated. For example:

```
profile:      @class
object type:  queue
entity:       test
entity type:  principal
authority:    crt
```

This shows that members of the group `test` have `crt` authority to the class `queue`.

3. For WebSphere MQ for Windows only, members of the "Administrators" group are by default given full authority. This authority, however, is given automatically by the OAM, and is not defined by the authority records. The dmpmqaut command displays authority defined only by the authority records. Unless an authority record has been explicitly defined, therefore, running the dmpmqaut command against the "Administrators" group will display no authority record for that group.

Related commands

dspmqaut	Display authority
setmqaut	Set or reset authority

Parent topic: The control commands

Related information
dspmqaut (display authority)
setmqaut (grant or revoke authority)

This build: January 26, 2011 11:55:28

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15670_

5.2.10. dmpmqlog

»Display and format a portion of the WebSphere MQ system log.«

Purpose

Use the dmpmqlog command to dump a formatted version of the WebSphere® MQ system log »to standard out«.

The log to be dumped must have been created on the same type of operating system as that being used to issue the command.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

ExtentNumber\n' + '3? -e *EndLSN*\n' + '4? -f *LogFilePath*\n' + '5? -m *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dmpmqlog--+--------------------+--+--------------+----------->\n'
+ '              +- -b ---------------+  \'- -e --EndLSN-\'    \n'
+ '              +- -s --StartLSN-----+                       \n'
+ '              \'- -n --ExtentNumber-\'                        \n'
+ '\n'
+ '>--+--------------------+--+----------------+-------------------><\n'
+ '    \'- -f --LogFilePath-\'  \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

Dump start point

Use one of the following parameters to specify the log sequence number (LSN) at which the dump should start. If you omit this, dumping starts by default from the LSN of the first record in the active portion of the log.

`-b`

Start dumping from the base LSN. The base LSN identifies the start of the log extent that contains the start of the active portion of the log.

`-s` *`StartLSN`*

Start dumping from the specified LSN. The LSN is specified in the format `nnnn:nnnn:nnnn:nnnn`.

If you are using a circular log, the LSN value must be equal to or greater than the base LSN value of the log.

`-n` *`ExtentNumber`*

Start dumping from the specified extent number. The extent number must be in the range 0–9 999 999.

This parameter is valid only for queue managers using linear logging.

`-e` *`EndLSN`*

End dumping at the specified LSN. The LSN is specified in the format `nnnn:nnnn:nnnn:nnnn`.

`-f` *`LogFilePath`*

The absolute (rather than relative) directory path name to the log files. The specified directory must contain the log header file (`amqhlctl.lfh`) and a subdirectory called `active`. The active subdirectory must contain the log files. By default, log files are assumed to be in the directories specified in the WebSphere MQ configuration information. If you use this option, queue names associated with queue identifiers are shown in the dump only if you use the -m option to name a queue manager name that has the object catalog file in its directory path.

On a system that supports long file names this file is called qmqmobjcat and, to map the queue identifiers to queue names, it must be the file used when the log files were created. For example, for a queue manager named qm1, the object catalog file is located in the directory `..\qmgrs\qm1\qmanager\`. To achieve this mapping, you might need to create a temporary queue manager, for example named tmpq, replace its object catalog with the one associated with the specific log files, and then start dmpmqlog, specifying `-m tmpq` and `-f` with the absolute directory path name to the log files.

`-m` *`QMgrName`*

The name of the queue manager. If you omit this parameter, the name of the default queue manager is used.

Note: Do not dump the log while the queue manager is running, and do not start the queue manager while dmpmqlog is running.

Parent topic: The control commands

This build: January 26, 2011 11:55:28

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15680_

5.2.11. dspmq

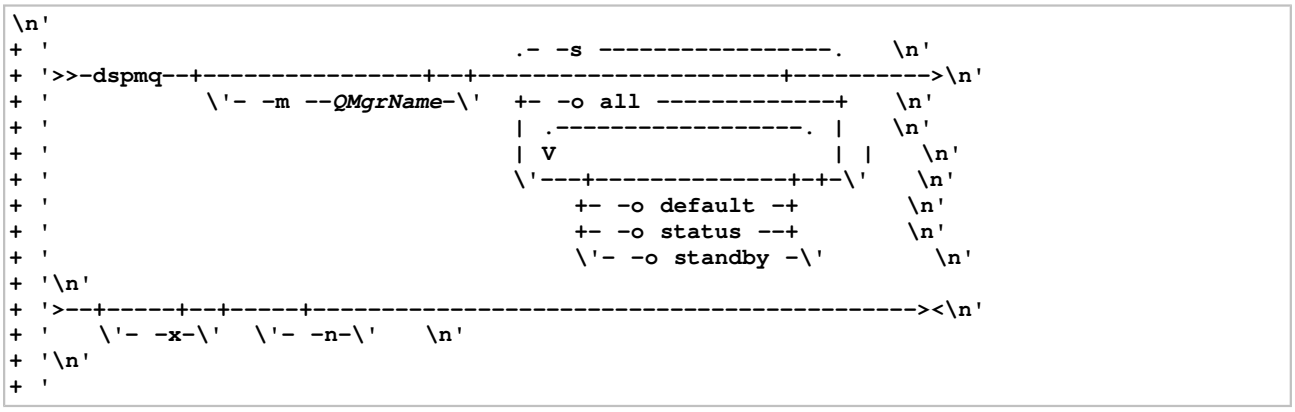
Display information about queue managers.

Purpose

Use the dspmq command to display names and details of the queue managers on a system.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('



');} //]]>

Required parameters

None

Optional parameters

-m *QMgrName*

The queue manager for which to display details. If you give no name, all queue manager names are displayed.

-n

Suppresses the translation of output strings.

-s

Displays the operational status of the queue managers. This is the default status setting.

The parameter *-o status* is equivalent to *-s*.

-o *all*

Displays the operational status of the queue managers, and whether any are the default queue manager.

-o *default*

Displays whether any of the queue managers are the default queue manager.

-o *status*

Displays the operational status of the queue managers.

-o *standby*

Displays whether a queue manager currently permits starting a standby instance. The possible values are shown in Table 1.

Table 1. Standby values

Permitted	The queue manager is running and is permitting standby instances.
Not permitted	The queue manager is running and is not permitting standby instances.
Not applicable	The queue manager is not running. You can start the queue manager and this instance will become active if it starts successfully.

-x

Displays information about instances of queue managers. The possible values are shown in Table 2.

Table 2. Instance values

Active	The instance is the active instance.
Standby	The instance is a standby instance.

Queue Manager States

The following is a list of the different states a queue manager can be in:

Starting
Running
Running as standby
Running elsewhere
Quiescing
Ending immediately
Ending preemptively
Ended normally
Ended immediately
Ended unexpectedly
Ended preemptively
Not available

Return codes

0	Command completed normally
36	Invalid arguments supplied
71	Unexpected error
72	Queue manager name error

Examples

Queue manager on this server that has ended immediately.

```
dspmq -o all
QMNAME(fred) STATUS(Ended immediately) DEFAULT(No) STANDBY(Not applicable)
```

Standby information for a queue manager on this server that has ended immediately.

```
dspmq -o standby
QMNAME(fred) STANDBY(Not applicable)
```

Standby information and instance information for a single instance queue manager started on this server.

```
dspmq -o standby -x
QMNAME(fred) STANDBY(Not permitted)
   INSTANCE(machine1) MODE(Active)
```

Instance information for a multi-instance queue manager started and active on this server.

```
dspmq -x
QMNAME(fred) STATUS(Running)
   INSTANCE(machine1) MODE(Active)
   INSTANCE(machine2) MODE(Standby)
```

Instance information for a multi-instance queue manager on standby on this server.

```
dspmq -x
QMNAME(fred) STATUS(Running as standby)
   INSTANCE(machine1) MODE(Active)
```

```
   INSTANCE(machine2) MODE(Standby)
```

Instance information for a multi-instance queue manager configured, but not started on this server.

```
dspmq -x
QMNAME(fred) STATUS(Running elsewhere)
   INSTANCE(machine1) MODE(Active)
   INSTANCE(machine2) MODE(Standby)
```

Parent topic: The control commands

This build: January 26, 2011 11:55:29

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15690_

5.2.12. dspmqaut

dspmqaut displays the authorizations of a specific WebSphere MQ object.

Purpose

Use the dspmqaut command to display the current authorizations to a specified object.

If a user ID is a member of more than one group, this command displays the combined authorizations of all the groups.

Only one group or principal can be specified.

For more information about authorization service components, see Installable services, Service components, and Authorization service.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

Profile -t *ObjectType*\n' + '4 -g *GroupName*\n' + '4 -p *PrincipalName*\n' + '5? -s *ServiceComponent*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqaut--+-----------------+-- -n --Profile------------------>\n'
+ '             \'- -m --QMgrName-\'                    \n'
+ '\n'
+ '>-- -t --ObjectType--+- -g --GroupName-----+-------------------->\n'
+ '                     \'- -p --PrincipalName-\'    \n'
+ '\n'
+ '>--+------------------------+-------------------------------><\n'
+ '    \'- -s --ServiceComponent-\'    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-n *Profile*

The name of the profile for which to display authorizations. The authorizations apply to all WebSphere® MQ objects with names that match the profile name specified. The profile name can be generic, using wildcard characters to specify a range of names as explained in Using OAM generic profiles on UNIX systems and Windows.

If you give an explicit profile name (without any wildcard characters), the object identified must exist.

This parameter is required, unless you are displaying the authorizations of a queue manager, in which case you *must not* include it and instead specify the queue manager name using the -m parameter.

-t *ObjectType*

The type of object on which to make the inquiry. Possible values are:

authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel

clntconn or **clcn**	A client connection channel
listener or **lstr**	A Listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter
qmgr	A queue manager
service or **srvc**	A service
topic or **top**	A topic

Optional parameters

-m *QMgrName*

The name of the queue manager on which to make the inquiry. This parameter is optional if you are displaying the authorizations of your default queue manager.

-g *GroupName*

The name of the user group on which to make the inquiry. You can specify only one name, which must be the name of an existing user group. On Windows systems, you can use only local groups.

-p *PrincipalName*

The name of a user for whom to display authorizations to the specified object.

For WebSphere MQ for Windows only, the name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see Principals and groups.

-s *ServiceComponent*

If installable authorization services are supported, specifies the name of the authorization service to which the authorizations apply. This parameter is optional; if you omit it, the authorization inquiry is made to the first installable component for the service.

Returned parameters

Returns an authorization list, which can contain none, one, or more authorization values. Each authorization value returned means that any user ID in the specified group or principal has the authority to perform the operation defined by that value.

Table 1 shows the authorities that can be given to the different object types.

Table 1. Specifying authorities for different object types

Authority	Queue	Process	Queue manager	Namelist	Topic	Auth info	Clntconn	Channel	List
all	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
alladm	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
allmqi	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
none	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
altusr	No	No	Yes	No	No	No	No	No	No
browse	Yes	No	No	No	No	No	No	No	No
chg	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
clr	Yes	No	No	No	Yes	No	No	No	No
connect	No	No	Yes	No	No	No	No	No	No
crt	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
ctrl	No	No	No	No	Yes	No	No	Yes	Yes
ctrlx	No	No	No	No	No	No	No	Yes	No
dlt	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
dsp	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
get	Yes	No	No	No	No	No	No	No	No
pub	No	No	No	No	Yes	No	No	No	No
put	Yes	No	No	No	Yes	No	No	No	No
inq	Yes	Yes	Yes	Yes	No	Yes	No	No	No
passall	Yes	No	No	No	Yes	No	No	No	No
passid	Yes	No	No	No	Yes	No	No	No	No
»resume«	No	No	No	No	Yes	No	No	No	No

set	Yes	Yes	Yes	No	No	No	No	No	No
setall	Yes	No	Yes	No	Yes	No	No	No	No
setid	Yes	No	Yes	No	Yes	No	No	No	No
sub	No	No	No	No	Yes	No	No	No	No
»system«	»No«	»No«	»Yes«	»No«	»No«	»No«	»No«	»No«	»No«

The following list defines the authorizations associated with each value:

»

all	Use all operations relevant to the object
alladm	Perform all administration operations relevant to the object
allmqi	Use all MQI calls relevant to the object
altusr	Specify an alternate user ID on an MQI call
browse	Retrieve a message from a queue by issuing an **MQGET** call with the BROWSE option
chg	Change the attributes of the specified object, using the appropriate command set
clr	Clear a queue (PCF command Clear queue only) or a topic
ctrl	Start, and stop the specified channel, listener, or service, and ping the specified channel.
ctrlx	Reset or resolve the specified channel
connect	Connect the application to the specified queue manager by issuing an **MQCONN** call
crt	Create objects of the specified type using the appropriate command set
dlt	Delete the specified object using the appropriate command set
dsp	Display the attributes of the specified object using the appropriate command set
get	Retrieve a message from a queue by issuing an **MQGET** call
inq	Make an inquiry on a specific queue by issuing an **MQINQ** call
passall	Pass all context
passid	Pass the identity context
»pub«	»Publish a message on a topic using the **MQPUT** call.«
put	Put a message on a specific queue by issuing an **MQPUT** call
»resume«	»Resume a subscription using the **MQSUB** call.«
set	Set attributes on a queue from the MQI by issuing an **MQSET** call
setall	Set all context on a queue
setid	Set the identity context on a queue
»sub«	»Create, alter or resume a subscription to a topic using the **MQSUB** call.«
»system«	»Use queue manager for internal system operations«

«

The authorizations for administration operations, where supported, apply to these command sets:

- **Control commands**
- **MQSC commands**
- **PCF commands**

Return codes

0	Successful operation
»26«	»Queue manager running as a standby instance.«
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
133	Unknown object name
145	Unexpected object name
146	Object name missing
147	Object type missing
148	Invalid object type
149	Entity name missing

Examples

- **The following example shows a command to display the authorizations on queue manager `saturn.queue.manager` associated with user group `staff`:**

  ```
  dspmqaut -m saturn.queue.manager -t qmgr -g staff
  ```

  The results from this command are:

  ```
  Entity staff has the following authorizations for object:
          get
          browse
          put
          inq
          set
          connect
          altusr
          passid
          passall
          setid
  ```

- **The following example displays the authorities user1 has for queue a.b.c:**

  ```
  dspmqaut -m qmgr1 -n a.b.c -t q -p user1
  ```

  The results from this command are:

  ```
  Entity user1 has the following authorizations for object:
          get
          put
  ```

Related commands

dmpmqaut	Dump authority
setmqaut	Set or reset authority

Parent topic: The control commands

Related information
dmpmqaut (dump authority)
setmqaut (grant or revoke authority)

This build: January 26, 2011 11:55:30

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15700_

5.2.13. dspmqcsv

Display the status of a command server

Purpose

Use the dspmqcsv command to display the status of the command server for the specified queue manager.

The status can be one of the following:

- **Starting**
- **Running**
- **Running with SYSTEM.ADMIN.COMMAND.QUEUE not enabled for gets**
- **Ending**
- **Stopped**

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

```
');break; default:document.write('

\n'
+ '>>-dspmqcsv--+----------+-------------------------------------><\n'
+ '              \'-QMgrName-\'    \n'
+ '\n'
+ '

');} //]]>
```

Required parameters

None

Optional parameters

QMgrName

The name of the local queue manager for which the command server status is being requested.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

Examples

The following command displays the status of the command server associated with `venus.q.mgr`:

```
dspmqcsv venus.q.mgr
```

Related commands

strmqcsv	Start a command server
endmqcsv	End a command server

Parent topic: The control commands

This build: January 26, 2011 11:55:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15710_

5.2.14. dspmqfls

Display the file names corresponding to WebSphere MQ objects.

Purpose

Use the dspmqfls command to display the real file system name for all WebSphere® MQ objects that match a specified criterion. You can use this command to identify the files associated with a particular object. This is useful for backing up specific objects. See Understanding WebSphere MQ file names for information about name transformation.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

ObjType\n' + '4 *GenericObjName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqfls--+----------------+--+---------------+-------------->\n'
+ '               \'- -m --QMgrName-\'   \'- -t --ObjType-\'    \n'
+ '\n'
+ '>--GenericObjName-------------------------------------------><\n'
+ '\n'
+ '
```

');} //]]>

Required parameters

GenericObjName

The name of the object. The name is a string with no flag and is a required parameter. Omitting the name returns an error.

This parameter supports an asterisk (*) as a wildcard at the end of the string.

Optional parameters

-m *QMgrName*

The name of the queue manager for which to examine files. If you omit this name, the command operates on the default queue manager.

-t *ObjType*

The object type. The following list shows the valid object types. The abbreviated name is shown first followed by the full name.

* or **all**	All object types; this is the default
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
catalog or **ctlg**	An object catalog
namelist or **nl**	A namelist
listener or **lstr**	A listener
process or **prcs**	A process
queue or **q**	A queue or queues matching the object name parameter
qalias or **qa**	An alias queue
qlocal or **ql**	A local queue
qmodel or **qm**	A model queue
qremote or **qr**	A remote queue
qmgr	A queue manager object
service or **srvc**	A service

Note:

1. **The dspmqfls command displays the name of the directory containing the queue, *not* the name of the queue itself.**
2. **In WebSphere MQ for UNIX systems, you need to prevent the shell from interpreting the meaning of special characters, for example, an asterisk (*). The way you do this depends on the shell you are using, but might involve the use of single quotation marks, double quotation marks, or a backslash.**

Return codes

0	Command completed normally
10	Command completed but not entirely as expected
20	An error occurred during processing

Examples

1. **The following command displays the details of all objects with names beginning `SYSTEM.ADMIN` defined on the default queue manager.**

```
dspmqfls SYSTEM.ADMIN*
```

2. **The following command displays file details for all processes with names beginning `PROC` defined on queue manager `RADIUS`.**

```
dspmqfls -m RADIUS -t prcs PROC*
```

Parent topic: The control commands

This build: January 26, 2011 11:55:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15720_

5.2.15. dspmqinf

Display WebSphere MQ configuration information (Windows and Unix platforms only).

Purpose

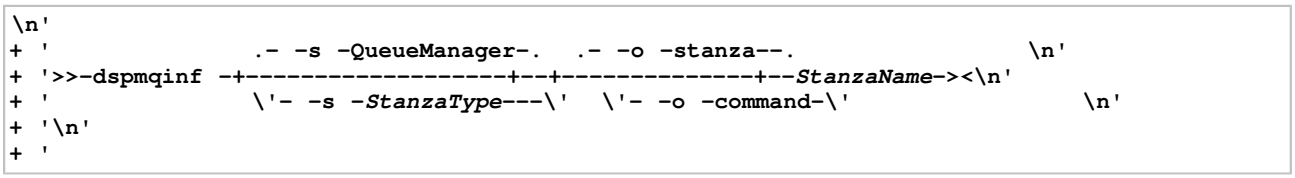
Use the dspmqinf command to display WebSphere MQ configuration information.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

StanzaName\n' + '\n' + '

');break; default:document.write('



');} //]]>

Required parameters

StanzaName

The name of the stanza. That is, the value of the key attribute that distinguishes between multiple stanzas of the same type.

Optional parameters

-s *StanzaType*

The type of stanza to display. If omitted, the QueueManager stanza is displayed.
The only supported value of *StanzaType* is `QueueManager`.

-o `stanza`

Displays the configuration information in stanza format as it appears in the `.ini` files. This is the default output format.
Use this format to display stanza information in a format that is easy to read.

-o `command`

Displays the configuration information as an addmqinf command.
Use this format to paste into a command shell.

Return codes

0	Successful operation
39	Bad command line parameters
44	Stanza does not exist
69	Storage not available
71	Unexpected error
72	Queue manager name error

Examples

```
dspmqinf QM.NAME
```

The command defaults to searching for a QueueManager stanza named `QM.NAME` and displays it in stanza format.

```
QueueManager:
    Name=QM.NAME
    Prefix=/var/mqm
    Directory=QM!NAME
    DataPath=/MQHA/qmgrs/QM!NAME
```

The following command gives the same result:

```
dspmqinf -s QueueManager -o stanza QM.NAME
```

The next example displays the output in addmqinf format.

```
dspmqinf -o command QM.NAME
```

The output is on one line:

```
addmqinf -s QueueManager -v Name=QM.NAME -v Prefix=/var/mqm -v Directory=QM!NAME
          -v DataPath=/MQHA/qmgrs/QM!NAME
```

Usage notes

Use dspmqinf in conjunction with addmqinf to create a new instance of a » multi-instance « queue manager on a different server.

To use this command you must be a WebSphere MQ administrator and a member of the `mqm` group.

Related commands

addmqinf	Add queue manager configuration information
rmvmqinf	Remove queue manager configuration information

Parent topic: The control commands

This build: January 26, 2011 11:55:30

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15701_

5.2.16. dspmqrte

Determine the route that a message has taken through a queue manager network.

Purpose

The WebSphere® MQ display route application (dspmqrte) can be executed on all platforms except z/OS®. You can execute the WebSphere MQ display route application as a client to a WebSphere MQ for z/OS queue manager by specifying the -c parameter when issuing the dspmqrte command.

Note: To run a client application against a WebSphere MQ for z/OS queue manager, the Client Attachment feature must be installed.

The WebSphere MQ display route application generates and puts a trace-route message into a queue manager network. As the trace-route message travels through the queue manager network, activity information is recorded. When the trace-route message reaches its target queue, the activity information is collected by the WebSphere MQ display route application and displayed. For more information, and examples of using the WebSphere MQ display route application, see the Monitoring WebSphere MQ book.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

TargetQName\n' + '5? -m *QMgrName*\n' + '\n' + 'Generation options\n' + '\n' + '1? -ac \n' + '1.1? -ar \n' + '2? -d *Deliver*\n' + '3? -f *Forward*\n' + '4? -l *Persistence* (1) \n' + '5? -o \n' + '6? -p *Priority*\n' + '7? -qm *TargetQMgrName*\n' + '8? \n' + '8 -ro \n' + '8.1 none \n' + '8.1+ ,\n' + '8.1 *ReportOption*\n' + '9? -rq *ReplyToQ*\n' + '9.1? -rqm *ReplyToQMgr*\n' + '10? -s *Activities*\n' + '11? -t *Detail*\n' + '12? -ts *TopicString*\n' + '13? -xp *PassExpiry*\n' + '14? -xs *Expiry*\n' + '15! %Display options\n' + '15?\n' + '15.0 -n (2) \n' + '\n' + 'Display options\n' + '\n' + '1? -b \n' + '2! -v summary \n' + '2?\n' + '2 -v \n' + '2.1 all \n' + '2.1 none \n' + '2.1 outline \n' + '2.1.1? \n' + '2.1.1+ ,\n' + '2.1.1 *DisplayOption* \n' + '3? -w *WaitTime*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqrte--+------+-----------------------------------------><\n'
+ '              \'- -c -\'    \n'
+ '\n'
+ '    .-| Generation options |--------------.                      \n'
+ '>--+-------------------------------------+-- -q --TargetQName---><\n'
+ '    \'- -i --CorrelId--| Display options |-\'                        \n'
+ '\n'
+ '>--+-----------------+-----------------------------------------><\n'
+ '    \'- -m --QMgrName-\'    \n'
+ '\n'
+ 'Generation options\n'
+ '\n'
+ '|--+------------------+--+---------------+--+---------------+--->\n'
+ '    \'- -ac --+-------+-\'  \'- -d --Deliver-\'  \'- -f --Forward-\'    \n'
+ '              \'- -ar -\'                                          \n'
+ '\n'
+ '>--+----------------------+--+------+--+---------------+------><\n'
+ '    |                      (1) |  \'- -o -\'  \'- -p --Priority-\'    \n'
+ '    \'- -l --Persistence-----\'                                    \n'
+ '\n'
+ '>--+----------------------+--+----------------------------+--->\n'
+ '    \'- -qm --TargetQMgrName-\'  \'- -ro --+- none -----------+-\'    \n'
+ '                                          | .-,------------. |      \n'
+ '                                          | V              | |      \n'
+ '                                          \'---ReportOption-+-\'      \n'
```

```
+ '\n'
+ '>--+-----------------------------------------+-----------------><\n'
+ '    \'- -rq --ReplyToQ--+---------------------+-\'    \n'
+ '                        \'- -rqm --ReplyToQMgr-\'      \n'
+ '\n'
+ '>--+------------------+--+--------------+-----------------------><\n'
+ '    \'- -s --Activities-\'  \'- -t --Detail-\'    \n'
+ '\n'
+ '>--+--------------------+--+------------------+-----------------><\n'
+ '    \'- -ts --TopicString-\'  \'- -xp --PassExpiry-\'    \n'
+ '\n'
+ '                         .-| Display options |-.    \n'
+ '>--+---------------+--+--------------------+-------------------|\n'
+ '    \'- -xs --Expiry-\'  |          (2)           |    \n'
+ '                         \'--- -n --------------\'    \n'
+ '\n'
+ 'Display options\n'
+ '\n'
+ '               .- -v  summary --------------------------------.    \n'
+ '|--+------+--+-------------------------------------------------+--><\n'
+ '    \'- -b -\'  \'- -v --+- all ------------------------------+-\'    \n'
+ '                        +- none -----------------------------+      \n'
+ '                        \'- outline --+---------------------+-\'      \n'
+ '                                      | .-,---------------. |        \n'
+ '                                      | V                 | |          \n'
+ '                                      \'--- DisplayOption -+-\'          \n'
+ '\n'
+ '>--+---------------+-----------------------------------------|\n'
+ '    \'- -w --WaitTime-\'    \n'
+ '\n'
+ '
```

');} //]]>

Notes:

1. If *Persistence* is specified as `yes`, and is accompanied by a request for a trace-route reply message (*-ar*), or any report generating options (*-ro ReportOption*), then you must specify the parameter *-rq ReplyToQ*. The reply-to queue must not resolve to a temporary dynamic queue.
2. If this parameter is accompanied by a request for a trace-route reply message (*-ar*), or any of the report generating options (*-ro ReportOption*), then a specific (non-model) reply-to queue must be specified using *-rq ReplyToQ*. By default, activity report messages are requested.

Required parameters

-q *TargetQName*

If the WebSphere MQ display route application is being used to send a trace-route message into a queue manager network, *TargetQName* specifies the name of the target queue.

If the WebSphere MQ display route application is being used to view previously gathered activity information, *TargetQName* specifies the name of the queue where the activity information is stored.

Optional parameters

-c

Specifies that the WebSphere MQ display route application connects as a client application. For more information on how to set up client machines, see the WebSphere MQ Clients book.

If you do not specify this parameter, the WebSphere MQ display route application does not connect as a client application.

-i *CorrelId*

This parameter is used when the WebSphere MQ display route application is used to display previously accumulated activity information only. There can be many activity reports and trace-route reply messages on the queue specified by *-q TargetQName*. *CorrelId* is used to identify the activity reports, or a trace-route reply message, related to a trace-route message. Specify the message identifier of the original trace-route message in *CorrelId*.

The format of *CorrelId* is a 48 character hexadecimal string.

-m *QMgrName*

The name of the queue manager to which the WebSphere MQ display route application connects. The name can contain up to 48 characters.

If you do not specify this parameter, the default queue manager is used.

Generation options

The following parameters are used when the WebSphere MQ display route application is used to put a trace-route message into a queue manager network.

-ac

Specifies that activity information is to be accumulated within the trace-route message.

If you do not specify this parameter, activity information is not accumulated within the trace-route message.

-ar

Requests that a trace-route reply message containing all accumulated activity information is generated in the following circumstances:

- **The trace-route message is discarded by a WebSphere MQ Version 7.0 queue manager.**
- **The trace-route message is put to a local queue (target queue or dead-letter queue) by a WebSphere MQ Version 7.0 queue manager.**
- **The number of activities performed on the trace-route message exceeds the value of specified in *-s Activities*.**

For more information on trace-route reply messages, see the Monitoring WebSphere MQ book.

If you do not specify this parameter, a trace-route reply message will not be requested.

-d *Deliver*

Specifies whether the trace-route message is to be delivered to the target queue on arrival. Possible values for *Deliver* are:

yes	On arrival, the trace-route message is put to the target queue, even if the queue manager does not support trace-route messaging.
no	On arrival, the trace-route message is **not** put to the target queue.

If you do not specify this parameter, the trace-route message is not put to the target queue.

-f *Forward*

Specifies the type of queue manager that the trace-route message can be forwarded to. Queue managers use an algorithm when determining whether to forward a message to a remote queue manager. For details of this algorithm, see Monitoring WebSphere MQ. The possible values for *Forward* are:

all	The trace-route message is forwarded to any queue manager. **Warning:** If forwarded to a WebSphere MQ queue manager prior to Version 6.0, the trace-route message will not be recognized and can be delivered to a local queue despite the value of the *-d Deliver* parameter.
supported	The trace-route message is only forwarded to a queue manager that will honor the *Deliver* parameter from the *TraceRoute* PCF group.

If you do not specify this parameter, the trace-route message will only be forwarded to a queue manager that will honor the *Deliver* parameter.

-l *Persistence*

Specifies the persistence of the generated trace-route message. Possible values for *Persistence* are:

yes	The generated trace-route message is persistent. (MQPER_PERSISTENT).
no	The generated trace-route message is **not** persistent. (MQPER_NOT_PERSISTENT).
q	The generated trace-route message inherits its persistence value from the queue specified by *-q TargetQName*. (MQPER_PERSISTENCE_AS_Q_DEF).

A trace-route reply message, or any report messages, returned will share the same persistence value as the original trace-route message.

If *Persistence* is specified as yes, you must specify the parameter *-rq ReplyToQ*. The reply-to queue must not resolve to a temporary dynamic queue.

If you do not specify this parameter, the generated trace-route message is not persistent.

-o

Specifies that the target queue is not bound to a specific destination. Typically this parameter is used when the trace-route message is to be put across a cluster. The target queue is opened with option MQOO_BIND_NOT_FIXED.

If you do not specify this parameter, the target queue is bound to a specific destination.

-p ***Priority***

Specifies the priority of the trace-route message. The value of *Priority* is either greater than or equal to 0, or MQPRI_PRIORITY_AS_Q_DEF. MQPRI_PRIORITY_AS_Q_DEF specifies that the priority value is taken from the queue specified by *-q TargetQName*.

If you do not specify this parameter, the priority value is taken from the queue specified by *-q TargetQName*.

-qm ***TargetQMgrName***

Qualifies the target queue name; normal queue manager name resolution will then apply. The target queue is specified with *-q TargetQName*.

If you do not specify this parameter, the queue manager to which the WebSphere MQ display route application is connected is used as the reply-to queue manager.

-ro **none** | ***ReportOption***

none	Specifies no report options are set.
ReportOption	Specifies report options for the trace-route message. Multiple report options can be specified using a comma as a separator. Possible values for *ReportOption* are:

activity

The report option MQRO_ACTIVITY is set.

coa

The report option MQRO_COA_WITH_FULL_DATA is set.

cod

The report option MQRO_COD_WITH_FULL_DATA is set.

exception

The report option MQRO_EXCEPTION_WITH_FULL_DATA is set.

expiration

The report option MQRO_EXPIRATION_WITH_FULL_DATA is set.

discard

The report option MQRO_DISCARD_MSG is set.

If neither *-ro ReportOption* nor *-ro none* are specified, then the MQRO_ACTIVITY and MQRO_DISCARD_MSG report options are specified.

-rq ***ReplyToQ***

Specifies the name of the reply-to queue that all responses to the trace-route message are sent to. If the trace-route message is persistent, or if the *-n* parameter is specified, a reply-to queue must be specified that is not a temporary dynamic queue.

If you do not specify this parameter, the system default model queue, SYSTEM.DEFAULT.MODEL.QUEUE is used as the reply-to queue. Using this model queue causes a temporary dynamic queue, for the WebSphere MQ display route application, to be created.

-rqm ***ReplyToQMgr***

Specifies the name of the queue manager where the reply-to queue resides. The name can contain up to 48 characters.

If you do not specify this parameter, the queue manager to which the WebSphere MQ display route application is connected is used as the reply-to queue manager.

-s ***Activities***

Specifies the maximum number of recorded activities that can be performed on behalf of the trace-route message before it is discarded. This prevents the trace-route message from being forwarded indefinitely if caught in an infinite loop. The value of *Activities* is either greater than or equal to 1, or MQROUTE_UNLIMITED_ACTIVITIES. MQROUTE_UNLIMITED_ACTIVITIES specifies that an unlimited number of activities can be performed on behalf of the trace-route message.

If you do not specify this parameter, an unlimited number of activities can be performed on behalf of the trace-route message.

-t ***Detail***

Specifies the activities that are recorded. The possible values for *Detail* are:

low	Activities performed by user-defined application are recorded only.
medium	Activities specified in **low** are recorded. Additionally, activities performed by MCAs are recorded.

high	Activities specified in **low**, and **medium** are recorded. MCAs do not expose any further activity information at this level of detail. This option is available to user-defined applications that are to expose further activity information only. For example, if a user-defined application determines the route a message takes by considering certain message characteristics, the routing logic could be included with this level of detail.

If you do not specify this parameter, medium level activities are recorded.

-ts ***TopicString***

Specifies a topic string to which the WebSphere MQ display route application is to publish a trace-route message, and puts this application into topic mode. In this mode, the application traces all of the messages that result from the publish request.

-xp ***PassExpiry***

Specifies whether the report option MQRO_DISCARD_MSG and the remaining expiry time from the trace-route message is passed on to the trace-route reply message. Possible values for *PassExpiry* are:

yes	The report option MQRO_PASS_DISCARD_AND_EXPIRY is specified in the message descriptor of the trace-route message. If a trace-route reply message, or activity reports, are generated for the trace-route message, the MQRO_DISCARD_MSG report option (if specified), and the remaining expiry time are passed on. This is the default value.
no	The report option MQRO_PASS_DISCARD_AND_EXPIRY is **not** specified. If a trace-route reply message is generated for the trace-route message, the discard option and remaining expiry time from the trace-route message are **not** passed on.

If you do not specify this parameter, the MQRO_PASS_DISCARD_AND_EXPIRY report option is not specified in the trace-route message.

-xs ***Expiry***

Specifies the expiry time for the trace-route message, in seconds.

If you do not specify this parameter, the expiry time is specified as 60 seconds.

-n

Specifies that activity information returned for the trace-route message is not to be displayed.

If this parameter is accompanied by a request for a trace-route reply message, (*-ar*), or any of the report generating options from (*-ro ReportOption*), then a specific (non-model) reply-to queue must be specified using *-rq ReplyToQ*. By default, activity report messages are requested.

After the trace-route message is put to the specified target queue, a 48 character hexadecimal string is returned containing the message identifier of the trace-route message. The message identifier can be used by the WebSphere MQ display route application to display the activity information for the trace-route message at a later time, using the *-i CorrelId* parameter.

If you do not specify this parameter, activity information returned for the trace-route message is displayed in the form specified by the *-v* parameter.

Display options

The following parameters are used when the WebSphere MQ display route application is used to display collected activity information.

-b

Specifies that the WebSphere MQ display route application will only browse activity reports or a trace-route reply message related to a message. This allows activity information to be displayed again at a later time.

If you do not specify this parameter, the WebSphere MQ display route application will destructively get activity reports or a trace-route reply message related to a message.

-v summary | all | none | outline *DisplayOption*

summary	The queues that the trace-route message was routed through are displayed.
all	All available information is displayed.
none	No information is displayed.
outline ***DisplayOption***	Specifies display options for the trace-route message. Multiple display options can be specified using a comma as a separator.

If no values are supplied the following is displayed:

- The application name
- The type of each operation
- Any operation specific parameters

Possible values for *DisplayOption* are:

activity

All non-PCF group parameters in *Activity* PCF groups are displayed.

identifiers

Values with parameter identifiers MQBACF_MSG_ID or MQBACF_CORREL_ID are displayed. This overrides *msgdelta*.

message

All non-PCF group parameters in *Message* PCF groups are displayed. When this value is specified, you cannot specify *msgdelta*.

msgdelta

All non-PCF group parameters in *Message* PCF groups, that have changed since the last operation, are displayed. When this value is specified, you cannot specify *message*.

operation

All non-PCF group parameters in *Operation* PCF groups are displayed.

traceroute

All non-PCF group parameters in *TraceRoute* PCF groups are displayed.

If you do not specify this parameter, a summary of the message route is displayed.

-w *WaitTime*

Specifies the time, in seconds, that the WebSphere MQ display route application will wait for activity reports, or a trace-route reply message, to return to the specified reply-to queue.

If you do not specify this parameter, the wait time is specified as the expiry time of the trace-route message, plus 60 seconds.

Return codes

0	Command completed normally
10	Invalid arguments supplied
20	An error occurred during processing

Examples

1. **The following command puts a trace-route message into a queue manager network with the target queue specified as `TARGET.Q`. Providing queue managers on route are enabled for activity recording, activity reports are generated. Depending on the queue manager attribute, ACTIVREC, activity reports are either delivered to the reply-to queue `ACT.REPORT.REPLY.Q`, or are delivered to a system queue. The trace-route message is discarded on arrival at the target queue.**

   ```
   dspmqrte -q TARGET.Q -rq ACT.REPORT.REPLY.Q
   ```

   Providing one or more activity reports are delivered to the reply-to queue, `ACT.REPORT.REPLY.Q`, the WebSphere MQ display route application orders and displays the activity information.

2. **The following command puts a trace-route message into a queue manager network with the target queue specified as `TARGET.Q`. Activity information is accumulated within the trace-route message, but activity reports are not generated. On arrival at the target queue the trace-route message is discarded. Depending on the value of the target queue manager attribute, ROUTEREC, a trace-route reply message can be generated and delivered to either the reply-to queue, `TRR.REPLY.TO.Q`, or to a system queue.**

   ```
   dspmqrte -ac -ar -ro discard -rq TRR.REPLY.TO.Q -q TARGET.Q
   ```

   Providing a trace-route reply message is generated and is delivered to the reply-to queue `TRR.REPLY.TO.Q`, the WebSphere MQ display route application orders and displays the activity information that was accumulated in the trace-route message.

For more examples of using the WebSphere MQ display route application and its output, see the Monitoring WebSphere MQ book.

Parent topic: The control commands

This build: January 26, 2011 11:55:32

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15730_

5.2.17. dspmqtrc

Format and display WebSphere MQ trace (Unix platforms only).

Purpose

The dspmqtrc command is supported on UNIX systems only. Use the dspmqtrc command to display WebSphere® MQ formatted trace output.

The runtime SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format any of the SSL trace files. The SSL trace files are binary files and, if they are transferred to IBM® support by FTP, they must be transferred in binary transfer mode.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
OutputFilename\n' + '6 *InputFileName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqtrc--+----------------------+--+-----+--+-----+---------->\n'
+ '              \'- -t --FormatTemplate-\'  \'- -h-\'  \'- -s-\'    \n'
+ '\n'
+ '>--+-----------------------+--InputFileName---------------------><\n'
+ '   \'- -o --OutputFilename-\'                    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

InputFileName

The name of the file containing the unformatted trace, for example:

```
/var/mqm/trace/AMQ12345.01.TRC
```

If you provide one input file, dspmqtrc formats it either to stdout or to the output file you name. If you provide more than one input file, any output file you name is ignored, and formatted files are named `AMQyyyyyy.zz.FMT`, based on the PID of the trace file.

Optional parameters

-t ***FormatTemplate***

The name of the template file containing details of how to display the trace. If this parameter is not supplied, the default template file location is used:

For AIX® systems, the default value is as follows:

```
/usr/mqm/lib/amqtrc2.fmt
```

For all UNIX systems other than AIX systems, the default value is as follows:

```
/opt/mqm/lib/amqtrc.fmt
```

-h

Omit header information from the report.

-s

Extract trace header and put to stdout.

-o ***output_filename***

The name of the file into which to write formatted data.

Related commands

endmqtrc	End trace
strmqtrc	Start trace

Parent topic: The control commands

This build: January 26, 2011 11:55:32

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15740_

5.2.18. dspmqtrn

Display in-doubt transactions.

Purpose

Use the dspmqtrn command to display details of in-doubt transactions. This includes transactions coordinated by WebSphere® MQ and by an external transaction manager.

For each in-doubt transaction, a transaction number (a human-readable transaction identifier), the transaction state, and the transaction ID are displayed. (Transaction IDs can be up to 128 characters long, hence the need for a transaction number.)

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

-i\n' + '4? *-m QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqtrn--+-----+--+-----+--+--------------+----------------><\n'
+ '              \'- -e-\'   \'- -i-\'   \'- -m QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-e

Requests details of externally coordinated, in-doubt transactions. Such transactions are those for which WebSphere MQ has been asked to prepare to commit, but has not yet been informed of the transaction outcome.

-i

Requests details of internally coordinated, in-doubt transactions. Such transactions are those for which each resource manager has been asked to prepare to commit, but WebSphere MQ has yet to inform the resource managers of the transaction outcome.

Information about the state of the transaction in each of its participating resource managers is displayed. This information can help you assess the affects of failure in a particular resource manager.

Note: If you specify neither -e nor -i, details of both internally and externally coordinated in-doubt transactions are displayed.

-m *QMgrName*

The name of the queue manager for which to display transactions. If you omit the name, the default queue manager's transactions are displayed.

Return codes

0	Successful operation
26	Queue manager running as a standby instance.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
102	No transactions found

Related commands

rsvmqtrn	Resolve transaction

Parent topic: The control commands

This build: January 26, 2011 11:55:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15750_

5.2.19. dspmqver

Display WebSphere MQ version and build information.

Purpose

Use the dspmqver command to display WebSphere® MQ version and build information.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

Fields\n' + '4? -b\n' + '5? -v\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-dspmqver--+------------------+--+--------------+--+-----+---->\n'
+ '              \'- -p --Components-\'   \'- -f --Fields-\'   \'- -b-\'    \n'
+ '\n'
+ '>--+-----+-------------------------------------------------------><\n'
+ '   \'- -v-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-p *Components*

Display information for the components specified by *Component*. Either a single component, or multiple components can be specified. To specify multiple components, sum the values of the required components, then specify *Component* as the total of the summation. Available components and related values follow:

1	WebSphere MQ server, or client.
2	WebSphere MQ classes for Java.
4	WebSphere MQ classes for Java Message Service.
8	WebScale Distribution Hub
16[1]	IBM WebSphere MQ custom channel for Windows Communication Foundation
32	IBM Message Service Client for .NET (XMS .NET)

The default value is 1.

Notes:

1. Supported by WebSphere MQ for Windows only. If you have not installed Microsoft .NET 3 or later, the following error message is displayed:
   ```
   Title: WMQWCFCustomChannelLevel.exe - Application Error
   The application failed to initialize properly (0x0000135).
   ```

-f *Fields*

Display information for the fields specified by *Field*. Either a single field, or multiple fields can be specified. To specify multiple fields, sum the values of the required fields, then specify *Field* as the total of the summation. Available fields and related values follow:

1	Name
2	Version, in the form `V.R.M.F`: Where `V`=Version, `R`=Release, `M`=Modification, and `F`=Fix pack
4	CMVC level
8	Build type

Information for each selected field is displayed on a separate line when the dspmqver command is run.

The default value is 15. This displays information for all fields.

-b

Omit header information from the report.

-v

Display verbose output.

Return codes

0	Command completed normally.
10	Command completed with unexpected results.
20	An error occurred during processing.

Examples

The following command displays WebSphere MQ version and build information, using the default settings for *-p Components* and *-f Fields*:

```
dspmqver
```

The following command displays version and build information for the WebSphere MQ classes for Java:

```
dspmqver -p 2
```

» The following command displays the Common Services for Java Platform Standard Edition, IBM WebSphere MQ, Java Message Service Client, and WebSphere MQ classes for Java Message Service:

```
dspmqver -p 4
```

«

The following command displays the build level of the WebScale Distribution Hub:

```
dspmqver -p 8 -f 4
```

» The following command displays the name and build type for IBM WebSphere MQ custom channel for Windows Communication Foundation:

```
dspmqver -p 16 -f 9
```

«

Parent topic: The control commands

This build: January 26, 2011 11:55:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15760_

5.2.20. endmqcsv

»Stop the command server for a queue manager.«

Purpose

Use the endmqcsv command to stop the command server on the specified queue manager.

If the queue manager attribute, SCMDSERV, is specified as QMGR then changing the state of the command server using endmqcsv does not effect how the queue manager acts upon the SCMDSERV attribute at the next restart.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

');break; default:document.write('

```
\n'
+ '               .- -c -.                \n'
+ '>>-endmqcsv--+------+--QMgrName--------------------------------><\n'
+ '               \'- -i -\'                \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

QMgrName

The name of the queue manager for which to end the command server.

Optional parameters

-c

Stops the command server in a controlled manner. The command server is allowed to complete the processing of any command message that it has already started. No new message is read from the command queue.

This is the default.

-i

Stops the command server immediately. Actions associated with a command message currently being processed might not complete.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

Examples

1. **The following command stops the command server on queue manager `saturn.queue.manager`:**

   ```
   endmqcsv  -c saturn.queue.manager
   ```

   The command server can complete processing any command it has already started before it stops. Any new commands received remain unprocessed in the command queue until the command server is restarted.

2. **The following command stops the command server on queue manager `pluto` immediately:**

   ```
   endmqcsv -i pluto
   ```

Related commands

strmqcsv	Start a command server
dspmqcsv	Display the status of a command server

Parent topic: The control commands

This build: January 26, 2011 11:55:33

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15770_

5.2.21. endmqlsr

End all listener process for a queue manager.

Purpose

The endmqlsr command ends all listener processes for the specified queue manager.

You do not need to stop the queue manager before issuing the endmqlsr command. If any of the listeners are configured to have inbound channels running within the runmqlsr listener process, rather than within a pool process, the request to end that listener might fail if channels are still active. In this case a message is written indicating how many listeners were successfully ended and how many listeners are still running.

If the listener attribute, CONTROL, is specified as QMGR then changing the state of the listener using endmqlsr does not effect how the queue manager acts upon the CONTROL attribute at the next restart.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
```

```
+ '>>-endmqlsr--+------+--+---------------+--------------------><\n'
+ '               \'- -w -\'   \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
');} //]]>
```

Optional parameters

-m *QMgrName*

The name of the queue manager. If you omit this, the command operates on the default queue manager.

-w

Wait before returning control.

Control is returned to you only after all listeners for the specified queue manager have stopped.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

Parent topic: The control commands

This build: January 26, 2011 11:55:34

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15780_

5.2.22. endmqdnm

Stop the .NET monitor for a queue (Windows only).

Purpose

The endmqdnm command applies to WebSphere® MQ for Windows only.

Use the endmqdnm control command to stop a .NET monitor.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-endmqdnm-- -q --QueueName--+----------------+--------------><\n'
+ '                               \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
');} //]]>
```

Required parameters

-q *QueueName*

The name of the application queue that the .NET monitor is monitoring.

Optional parameters

-m *QMgrName*

The name of the queue manager that hosts the application queue.

If omitted, the default queue manager is used.

Return codes

0	Successful operation

36	Invalid arguments supplied
40	Queue manager not available
71	Unexpected error
72	Queue manager name error
133	Unknown object name error

Parent topic: The control commands

This build: January 26, 2011 11:55:34

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15790_

5.2.23. endmqm

»Stop a queue manager or switch to a standby queue manager.«

Purpose

»Use the endmqm command to end (stop) a specified queue manager. This command stops a queue manager in one of three modes:

- **Controlled or quiesced shutdown**
- **Immediate shutdown**
- **Preemptive shutdown**

«

»The endmqm command stops all instances of a multi-instance queue manager in the same way as it stops a single instance queue manager. You can issue the endmqm on either the active instance, or one of the standby instances of a multi-instance queue manager. You must issue endmqm on the active instance to end the queue manager.«

»If you issue the endmqm command on the active instance of a multi-instance queue manager, you can permit a standby instance to switch over to being the new active instance when the current active instance completes its shutdown. «

»If you issue the endmqm command on a standby instance of a multi-instance queue manager, you can end the standby instance by adding the `-x` option, and leave the active instance running. The queue manager reports an error if you issue endmqm on the standby instance without the `-x` option.«

Issuing the endmqm command will affect any client application connected through a server-connection channel. The effect varies depending on the parameter used, but it is as though a STOP CHANNEL command was issued in one of the three possible modes. See WebSphere MQ Clients, for information on the effects of STOP CHANNEL modes on server-connection channels. The endmqm optional parameter descriptions state which STOP CHANNEL mode they will be equivalent to.

»If you issue endmqm to stop a queue manager, reconnectable clients do not try to reconnect. To override this behavior, specify either the `-r` or `-s` option to enable clients to start trying to reconnect.

Note: If a queue manager or a channel ends unexpectedly, reconnectable clients start trying to reconnect.

Note: The client might not reconnect to this queue manager. Depending on the MQCONNX reconnect option the client has used, and the definition of the queue manager group in the client connection table, the client might reconnect to a different queue manager. You can configure the client to force it to reconnect to the same queue manager.

«

The attributes of the queue manager and the objects associated with it are not affected by the endmqm command. You can restart the queue manager using the strmqm (Start queue manager) command.

To delete a queue manager, stop it and then use the dltmqm (Delete queue manager) command.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
+ '                          .- -c -.                                  \n'
+ '>>-endmqm--+------+--+-+------+--+------+-+--QMgrName----------><\n'
+ '            \'- -z -\'   | +- -w -+  +- -s -+ |                    \n'
+ '                         | +- -i -+  \'- -r -\' |                   \n'
+ '                         | \'- -p -\'           |                    \n'
```

```
+ '                          \'- -x ----------------\'               \n'
+ '\n'
+ '
');} //]]>
```

Required parameters

QMgrName

The name of the message queue manager to be stopped.

Optional parameters

-c

Controlled (or quiesced) shutdown. This is the default.

The queue manager stops, but only after all applications have disconnected. Any MQI calls currently being processed are completed.

Control is returned to you immediately and you are not notified of when the queue manager has stopped.

The effect on any client applications connected through a server-connection channel is equivalent to a STOP CHANNEL command issued in QUIESCE mode.

-i

Immediate shutdown. The queue manager stops after it has completed all the MQI calls currently being processed. Any MQI requests issued after the command has been issued fail. Any incomplete units of work are rolled back when the queue manager is next started.

Control is returned after the queue manager has ended.

The effect on any client applications connected through a server-connection channel is equivalent to a STOP CHANNEL command issued in FORCE mode.

-p

Preemptive shutdown.

***Use this type of shutdown only in exceptional circumstances.* For example, when a queue manager does not stop as a result of a normal endmqm command.**

The queue manager might stop without waiting for applications to disconnect or for MQI calls to complete. This can give unpredictable results for WebSphere® MQ applications. The shutdown mode is set to *immediate shutdown*. If the queue manager has not stopped after a few seconds, the shutdown mode is escalated, and all remaining queue manager processes are stopped.

The effect on any client applications connected through a server-connection channel is equivalent to a STOP CHANNEL command issued in TERMINATE mode.

» **-r** «

» **Start trying to reconnect reconnectable clients. This has the effect of reestablishing the connectivity of clients to other queue managers in their queue manager group.** «

» **-s** «

» **Switch over to a standby queue manager instance after shutting down. The commands checks there is a standby instance running before ending the active instance, but it does not wait for the standby instance to start before ending.**

Connections to the queue manager are broken by the active instance shutting down. Reconnectable clients start trying to reconnect.

You can configure the reconnection options of a client to reconnect only to another instance of the same queue manager, or to reconnect to other queue managers in the queue manager group.

«

-w

Wait shutdown.

This type of shutdown is equivalent to a controlled shutdown except that control is returned to you only after the queue manager has stopped. You receive the message `Waiting for queue manager qmName to end` while shutdown progresses.

The effect on any client applications connected through a server-connection channel is equivalent to a STOP CHANNEL command issued in QUIESCE mode.

-x

End a standby instance of the queue manager, without ending the active instance of the queue manager.

-z

Suppresses error messages on the command.

Return codes

0	Queue manager ended
3	Queue manager being created
16	Queue manager does not exist
40	Queue manager not available
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
77	WebSphere MQ queue manager cannot switch over
79	Active instance of WebSphere MQ queue manager *QmgrName* not ended
90	Standby instance of WebSphere MQ queue manager *QmgrName* not ended
119	Permission denied

Examples

The following examples show commands that stop the specified queue managers.

1. **This command ends the queue manager named `mercury.queue.manager` in a controlled way. All applications currently connected are allowed to disconnect.**

   ```
   endmqm mercury.queue.manager
   ```

2. **This command ends the queue manager named `saturn.queue.manager` immediately. All current MQI calls complete, but no new ones are allowed.**

   ```
   endmqm -i saturn.queue.manager
   ```

The results of issuing endmqm to the local instance of a multi-instance queue manager are shown in Table 1. The results of the command depend on whether the -s or -x switch is used, and the running status of local and remote instances of the queue manager.

Table 1. endmqm actions

endmqm option	Local machine	Remote machine	RC	Message	Result
	Active	None	0	-	Queue manager ended.
		Standby			Queue manager ended, including the standby instance.
	Standby	Active	90	AMQ8368	Standby instance of WebSphere MQ queue manager *QmgrName* not ended.
-s	Active	None	77	AMQ7276	WebSphere MQ queue manager cannot switch over.
		Standby	0	-	Queue manager QMNAME ended, permitting switchover to a standby instance.
	Standby	Active	90	AMQ8368	Standby instance of WebSphere MQ queue manager *QmgrName* not ended.
-x	Active	None	79	AMQ8367	Active instance of WebSphere MQ queue manager *QmgrName* not ended.
		Standby			
	Standby	Active	0	-	Standby instance of queue manager QMNAME ended.

Related commands

crtmqm	Create a queue manager
strmqm	Start a queue manager
dltmqm	Delete a queue manager

Parent topic: The control commands

This build: January 26, 2011 11:55:34

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15800_

5.2.24. endmqtrc

»End trace for some or all of the entities that are being traced.«

Purpose

Use the endmqtrc command to end tracing for the specified entity or all entities. The endmqtrc command ends only the trace that is described by its parameters. Using endmqtrc with no parameters ends early tracing of all processes.

Syntax

The syntax of this command is as follows:
Syntax diagram format ○ Railroad diagram ○ Dotted decimal
PidTids\n' + '4? -p *Apps*\n' + '5? -e \n' + '6? -a \n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-endmqtrc--+----------------+--+---------------+-------------->\n'
+ '              \'- -m --QMgrName-\'   \'- -i --PidTids-\'    \n'
+ '\n'
+ '>--+------------+--+------+--+------+------------------------><\n'
+ '    \'- -p --Apps-\'   \'- -e -\'   \'- -a -\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-m *QMgrName*

The name of the queue manager for which to end tracing.

» The QMgrName supplied must match exactly the QMgrName supplied on the strmqtrc command. If the strmqtrc command used wildcards, the endmqtrc command must use the same wildcard specification including the escaping of any wildcard characters to prevent them being processed by the command environment. «

A maximum of one -m flag and associated queue manager name can be supplied on the command.

-i *PidTids*

Process identifier (PID) and thread identifier (TID) for which to end tracing. You cannot use the -i flag with the -e flag. If you try to use the -i flag with the -e flag, then an error message is issued. This parameter should be used only under the guidance of IBM® Service personnel.

-p *Apps*

The named processes for which to end tracing. *Apps* is a comma-separated list. You must specify each name in the list exactly as the program name would be displayed in the "Program Name" FDC header. Asterisk (*) or question mark (?) wildcards are allowed. You cannot use the -p flag with the -e flag. If you try to use the -p flag with the -e flag, then an error message is issued.

-e

Ends early tracing of all processes.

Using endmqtrc with no parameters has the same effect as endmqtrc -e. You cannot specify the -e flag with the -m flag, the -i flag or the -p flag.

-a

Ends all tracing.

This flag *must* be specified alone.

Return codes

AMQ5611	This message is issued if you supply invalid arguments to the command.

Examples

This command ends tracing of data for a queue manager called `QM1`.

```
endmqtrc -m QM1
```

The following examples are a sequence that shows how the endmqtrc command ends only the trace that is described by its parameters.

1. **The following command enables tracing for queue manager QM1 and process amqxxx.exe:**

   ```
   strmqtrc -m QM1 -p amqxxx.exe
   ```

2. **The following command enables tracing for queue manager QM2:**

   ```
   strmqtrc -m QM2
   ```

3. **The following command ends tracing for queue manager QM2 only. Tracing of queue manager QM1 and process amqxxx.exe continues:**

   ```
   endmqtrc -m QM2
   ```

Related commands

dspmqtrc	Display formatted trace output
strmqtrc	Start trace

Parent topic: The control commands

This build: January 26, 2011 11:55:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15810_

5.2.25. migmbbrk

The migmbbrk command migrates publish/subscribe configuration data from WebSphere® Event Broker Version 6.0 or WebSphere Message Broker Version 6.0 or 6.1 to WebSphere MQ Version 7.0.1 or later versions.

Purpose

The migmbbrk command is not supported on all of the platforms that WebSphere MQ Version 7.0 supports. See "Supported operating systems" for details.

To use the migmbbrk command you must be using at least WebSphere Message Broker Version 6.0, Fix Pack 9, or WebSphere Message Broker Version 6.1, Fix Pack 4.

Use the migmbbrk command to migrate the publish/subscribe configuration data from a WebSphere Event Broker Version 6.0 or a WebSphere Message Broker Version 6.0 or Version 6.1 broker to a WebSphere MQ Version 7.0.1 or later queue manager. The command runs a migration process that migrates the following publish/subscribe configuration data to the queue manager that is associated with the named broker:

- **Subscriptions**
- **Subscription points. (Subscription points are supported only when RFH2 messages are used.)**
- **Streams**
- **Retained publications**

The migmbbrk command does not migrate the Access Control List (ACL). Instead, running the migration with the -t or -r parameters produces a file containing suggested setmqaut commands to set up a security environment in the queue manager that is equivalent to the security environment that existed in the broker. You must review and modify the security command file as needed and run the commands to set up a security environment in the queue manager, equivalent to the one that existed in the broker, before you run the migration with the -c parameter to complete the migration.

Note: On UNIX systems, all authorities are held by user groups internally, not by principals. This has the following implications:

- **If you use the setmqaut command to grant an authority to a principal, the authority is actually granted to the primary user group of the principal. This means that the authority is effectively granted to all members of that user group.**
- **If you use the setmqaut command to revoke an authority from a principal, the authority is actually revoked from the primary user group of the principal. This means that the authority is effectively revoked from all members of**

that user group.

You must issue the migmbbrk command from a command window that can execute both WebSphere MQ and WebSphere Message Broker commands successfully. Typically this is true if the command is issued from a WebSphere Message Broker command console.

The WebSphere Event Broker Version 6.0 or WebSphere Message Broker Version 6.0 or 6.1 publish/subscribe configuration data, which is stored in the subscription database tables, is not deleted by the migration process. This configuration data is therefore available to use until you explicitly delete it.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

');break; default:document.write('

```
\n'
+ '                          .----------.                          \n'
+ '                          V          |                           \n'
+ '>>-migmbbrk--+- -c -+----+------+-+-- -b --BrokerName----------><\n'
+ '                +- -r -+     +- -l -+                          \n'
+ '                \'- -t -\'     +- -o -+                           \n'
+ '                             +- -s -+                          \n'
+ '                             \'- -z -\'                           \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-b ***BrokerName***

The name of the broker that is the source of the publish/subscribe configuration data that is to be migrated. The queue manager to which the publish/subscribe configuration data is migrated is the queue manager that is associated with the named broker.

-c

Complete the migration of the publish/subscribe configuration data. The completion phase of the migration uses the topic objects that are created in the initial -t phase. It is possible that the broker state has changed since the initial phase was run and that new additional topic objects are now required. If this is the case, the completion phase creates these new topic objects as necessary. The completion phase does not delete any topic objects that have become unnecessary; you might need to delete any topic objects that you do not require.

Before you complete the migration you must review and modify the security command file produced in the -r or -t phase as required and execute the commands to set up a security environment in the queue manager, equivalent to the one that existed in the broker.

Before you run this completion phase you must run the initial -t phase. You cannot use the -c parameter with the -r parameter or the -t parameter. This phase also creates a migration log.

-r

Rehearse the migration process but do not change anything. You can use this before running the migration with the -t parameter, to create a migration log, including any errors, so that you can observe what the result of the migration process would be, but without changing the current configurations.

Rehearsing the migration also produces a file containing suggested setmqaut commands to set up a security environment in the queue manager that is equivalent to the security environment that existed in the broker. Before you complete the migration with the -c parameter you must review and modify the security command file as required and execute the commands to set up a security environment in the queue manager, equivalent to the one that existed in the broker.

You cannot use the -r parameter with the -c parameter or the -t parameter.

-t

Create topic objects that might be needed in the queue manager, based on the ACL entries that are defined in the broker.

Use of the -t parameter also produces a file containing suggested setmqaut commands to set up a security environment in the queue manager that is equivalent to the security environment that existed in the broker. The topic objects are created in anticipation of you executing the security commands to create ACLs for the topic objects. Before you complete the migration with the -c parameter you must review and modify the security command file as required and execute the commands to set up a security environment in the queue manager, equivalent to the one that existed in the broker.

You must run this phase before you run the completion phase with the -c parameter. You cannot use the -t parameter with the -c parameter or the -r parameter. This phase also creates a migration log.

Optional parameters

-l

Leave the broker running. If you do not specify this parameter, the broker is shut down by default at the end of the migration process.

-o

Overwrite any subscription or retained publication that already exists in the queue manager and that has the same name as a subscription or retained publication that is being migrated from the broker, with the publish/subscribe configuration data that was retrieved from the broker. The -o parameter has no effect if you use it with the -r parameter.

-s

Discard any intermediate configuration data that was retained from a previous instance of the migration process that failed or was interrupted. The migration process populates private queues with temporary data. If the migration process completes successfully, the temporary data is deleted. If you do not specify this parameter and the migration process fails or is interrupted, the temporary data is retained and is used by the migration process if you restart it, so that the process resumes at the point where it previously failed or was interrupted.

-z

Run the migration process, regardless of whether it has previously run to a successful completion. If you do not specify this parameter and the migration process has previously run to a successful completion, the process recognizes this fact and exits. You can use the -o parameter with the -z parameter, but this is not mandatory. A previous rehearsal of the migration using the -r parameter does not count as a successful completion.

Return codes

0	Migration completed successfully
20	An error occurred during processing

Output files

The migration process writes two output files to the current directory:

amqmigrateacl.txt

A file containing a list of setmqaut commands, created in the current directory for you to review, change, and run if appropriate, to help you to reproduce your ACLs.

amqmigmbbrk.log

A log file containing a record of the details of the migration.

Examples

This command migrates the publish/subscribe configuration data of broker BRK1 into its associated queue manager and specifies that the migration process runs regardless of whether it has previously run to a successful completion. It also specifies that any subscription or retained publication that already exists in the queue manager, that has the same name as a subscription or retained publication that is being migrated from the broker, must be overwritten.

```
migmbbrk -z -o -b BRK1
```

Supported operating systems

The migmbbrk command is supported only on the following platforms that support WebSphere Event Broker Version 6.0 or WebSphere Message Broker Version 6.0:

Microsoft Windows XP Professional with SP2, 32–bit versions only

Solaris x86-64 platform: Solaris 10

Solaris SPARC platform: Sun Solaris 9 (64–bit)

AIX Version 5.2 or later, 64–bit only

HP-UX PA-RISC platform: HP-UX 11i

HP-UX Itanium platform: HP-UX 11i

Linux zSeries (64-bit)

Linux PowerPC (64–bit)

Linux Intel x86

Linux Intel x86–64

On z/OS the equivalent function to the migmbbrk command is provided by the CSQUMGMB utility.

Parent topic: The control commands

Related reference
setmqaut

Related information
Publish/subscribe migration from WebSphere® Event Broker Version 6.0 or WebSphere Message Broker Version 6.0 to WebSphere MQ Version 7.0
Access Control List (ACL) migration

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23380_

5.2.26. mqftapp

Start the graphical interface to File Transfer (Windows and Linux x86 platforms only).

Purpose

The mqftapp command is available with the File Transfer Application on WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform) servers only.

Use the mqftapp command to run the File Transfer Application graphical user interface (GUI).

Alternatively, on WebSphere MQ for Windows you can start the File Transfer Application by selecting it through the start menu.

When run for the first time, the graphical user interface must be configured. For instructions of how to do this, see Configuring the GUI.

Syntax

The syntax of this command follows:
Syntax diagram format ○ Railroad diagram ○ Dotted decimal

\n' + '>>-mqftapp----------><\n' + '\n' + '

');} //]]>

Related commands

mqftrcv	Receive file on server
mqftrcvc	Receive file on client
mqftsnd	Send file from server
mqftsndc	Send file from client

Parent topic: The control commands

This build: January 26, 2011 11:55:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15820_

5.2.27. mqftrcv

Process files received on a server using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).

Purpose

The mqftrcv command is available with the File Transfer Application on WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform) servers only.

Use the mqftrcv command to do one of the following:

- **Receive a file.**
- **Extract a file.**

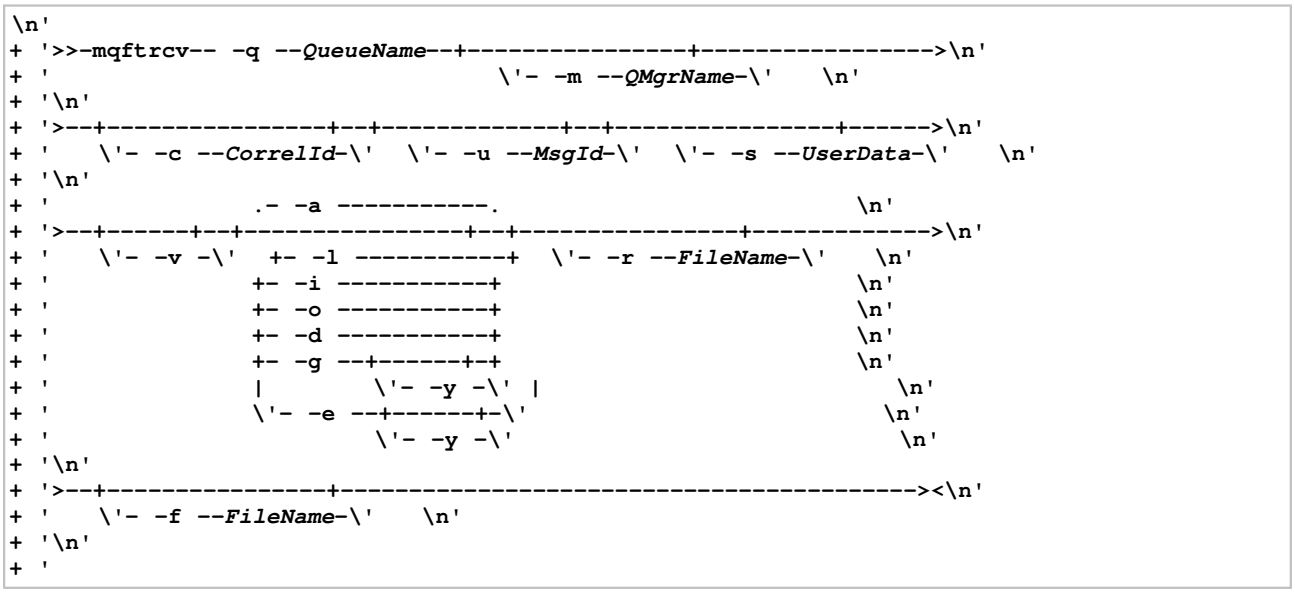
- Delete a file.
- View sent files.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '3? -c *CorrelId*\n' + '4? -u *MsgId*\n' + '5? -s *UserData*\n' + '6? -v \n' + '7! -a \n' + '7?\n' + '7 -l \n' + '7 -i \n' + '7 -o \n' + '7 -d \n' + '7 -g \n' + '7.1? -y \n' + '7 -e \n' + '7.1? -y \n' + '8? -r *FileName*\n' + '9? -f *FileName*\n' + '\n' + '

');break; default:document.write('



');} //]]>

Required parameters

-q *QueueName*

The local name of the destination queue.

Optional parameters

-m *QMgrName*

The name of the queue manager that hosts the destination queue. A queue manager that does not have the File Transfer Application installed can be specified. If you omit this parameter, the default queue manager is used.

-c *CorrelId*

Select all files matching *CorrelId*. Selection can be combined with *-s UserData*, and *-f FileName*.

-u *MsgID*

Select the message that has a message ID that matches *MsgID*. Used to select other messages.

-s *UserData*

Select files by locating any occurrence of the character string *UserData*, in part or all of the file's UserData. The comparison is case sensitive, and wildcard characters cannot be used.

Selection can be combined with *-c CorrelId*, and *-f FileName*.

-v

Return the *CorrelId*, and *MsgId* of the file.

-a

List all files and messages, in the following order:

1. Complete files, ordered by queue name
2. Incomplete files, ordered by queue name
3. Other messages, ordered by queue name

This is the default value. For more information on file status see File status.

-l

List all complete files, ordered by queue name.

-i

List all incomplete files, ordered by queue name.

-o

List all other messages, ordered by queue name.

-d

Delete the specified file, or the group of messages. If more than one file matches the selection criteria, no files are deleted and a return code is returned.

-g

Receive a complete file. Message associated with the file are removed. If a file already exists of the same name, do one of the following:

- **Specify the *-y* parameter, so that the existing file is overwritten.**
- **Specify the *-r FileName* parameter, so that the file is renamed.**

-e

Extract a complete, or incomplete file. Messages associated with the file are not removed. If a file already exists of the same name, do one of the following:

- **Specify the *-y* parameter, so that the existing file is overwritten.**
- **Specify the *-r FileName* parameter, so that the file is renamed.**

-y

Replace an existing file of the same name. Used with optional parameters *-g*, and *-e*.

-r *FileName*

Assign new file name and/or file location.

Used to rename, or to relocate a file. The file is assigned the name specified in *FileName*. A fully qualified file name can be specified to relocate the file. If the file name, or path, contains embedded spaces, it must be specified in double quotes. One file can be specified only, and you cannot use wildcard characters.

-f *FileName*

Select all files matching *FileName*. The fully qualified file name can be specified. If the file name contains embedded spaces, it must be specified in double quotes. You cannot use wildcard characters.

Selection can be combined with *-c CorrelId*, and *-s UserData*.

Return codes

0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
69	Storage not available
71	Unexpected error
163	Queue name required
164	Cannot open queue
165	Cannot open file
166	Cannot put to queue
167	No file name specified (Send)
168	Message length is too small to send data
169	Sending file has changed
170	Cannot get from queue
171	Cannot write to file
172	CorrelId is invalid
173	MsgId is invalid
174	No messages to receive
175	File for delete is not unique

Examples

This command lists all files and messages on the queue, `MY.QUEUE`, located on the default queue manager:

```
mqftrcv -q MY.QUEUE -a
```

This command gets the first complete file on the queue, `MY.QUEUE`, located on queue manager `QM1`:

```
mqftrcv -q MY.QUEUE -m QM1 -g
```

This command gets the complete file, named `My document.txt`, on the queue, `MY.QUEUE`, located on the default queue manager:

```
mqftrcv -q MY.QUEUE -g -f "My document.txt"
```

This command gets the complete file, named `My document.txt`, also marked `URGENT`, on the queue, `MY.QUEUE`, located on queue manager `QM1` :

```
mqftrcv -q MY.QUEUE -m QM1 -g -f "My document.txt" -s "URGENT"
```

Related commands

mqftapp	Run File Transfer Application
mqftrcvc	Receive file on client
mqftsnd	Send file from server
mqftsndc	Send file from client

Parent topic: The control commands

This build: January 26, 2011 11:55:35

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15830_

5.2.28. mqftrcvc

Process files received on a client (Windows and Linux x86 platforms only).

Purpose

The mqftrcvc command is available with the File Transfer Application on WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform) clients only.

Use the mqftrcvc command to do one of the following:

- **Receive a file from a connected server.**
- **Extract a file from a connected server.**
- **Delete a file from a connected server.**
- **View sent files on a connected server.**

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '3? -c *CorrelId*\n' + '4? -u *MsgId*\n' + '5? -s *UserData*\n' + '6? -v \n' + '7! -a \n' + '7?\n' + '7 -l \n' + '7 -i \n' + '7 -o \n' + '7 -d \n' + '7 -g \n' + '7.1? -y \n' + '7 -e \n' + '7.1? -y \n' + '8? -r *FileName*\n' + '9? -f *FileName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-mqftrcvc-- -q --QueueName--+----------------+---------------->\n'
+ '                               \'- -m --QMgrName-\'    \n'
+ '\n'
+ '>--+----------------+--+-------------+--+---------------+------>\n'
+ '    \'- -c --CorrelId-\'  \'- -u --MsgId-\'  \'- -s --UserData-\'    \n'
+ '\n'
+ '               .- -a -----------.                          \n'
+ '>--+------+--+----------------+--+---------------+-------------->\n'
+ '    \'- -v -\'  +- -l -----------+  \'- -r --FileName-\'    \n'
+ '               +- -i -----------+                          \n'
+ '               +- -o -----------+                          \n'
+ '               +- -d -----------+                          \n'
+ '               +- -g --+------+-+                          \n'
+ '               |       \'- -y -\' |                           \n'
+ '               \'- -e --+------+-\'                          \n'
+ '                        \'- -y -\'                           \n'
+ '\n'
+ '>--+----------------+-------------------------------------------><\n'
+ '    \'- -f --FileName-\'    \n'
+ '\n'
```

```
+ '
');} //]]>
```

Required parameters

-q *QueueName*

The local name of the destination queue.

Optional parameters

-m *QMgrName*

The name of the queue manager that hosts the destination queue. A queue manager that does not have the File Transfer Application installed can be specified. If you omit this parameter, the default queue manager is used.

-c *CorrelId*

Select all files matching *CorrelId*. Selection can be combined with *-s UserData*, and *-f FileName*.

-u *MsgID*

Select the message that has a message ID that matches *MsgID*. Used to select other messages.

-s *UserData*

Select files by locating any occurrence of the character string *UserData*, in part or all of the file's UserData. The comparison is case sensitive, and wildcard characters cannot be used.

Selection can be combined with *-c CorrelId*, and *-f FileName*.

-v

Return the *CorrelId*, and *MsgId* of the file.

-a

List all files and messages, in the following order:

1. **Complete files, ordered by queue name**
2. **Incomplete files, ordered by queue name**
3. **Other messages, ordered by queue name**

This is the default value. For more information on file status see File status.

-l

List all complete files, ordered by queue name.

-i

List all incomplete files, ordered by queue name.

-o

List all other messages, ordered by queue name.

-d

Delete the specified file, or the group of messages. If more than one file matches the selection criteria, no files are deleted and a return code is returned.

-g

Receive a complete file. Message associated with the file are removed. If a file already exists of the same name, do one of the following:

- **Specify the *-y* parameter, so that the existing file is overwritten.**
- **Specify the *-r FileName* parameter, so that the file is renamed.**

-e

Extract a complete, or incomplete file. Messages associated with the file are not removed. If a file already exists of the same name, do one of the following:

- **Specify the *-y* parameter, so that the existing file is overwritten.**
- **Specify the *-r FileName* parameter, so that the file is renamed.**

-y

Replace an existing file of the same name. Used with optional parameters *-g*, and *-e*.

-r *FileName*

Assign new file name and/or file location.

Used to rename, or to relocate a file. The file is assigned the name specified in *FileName*. A fully qualified file name can be specified to relocate the file. If the file name, or path, contains embedded spaces, it must be specified in double quotes. One file can be specified only, and you cannot use wildcard characters.

-f *FileName*

Select all files matching *FileName*. The fully qualified file name can be specified. If the file name contains embedded spaces, it must be specified in double quotes. You cannot use wildcard characters.

Selection can be combined with *-c CorrelId*, and *-s UserData*.

Return codes

0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
69	Storage not available
71	Unexpected error
163	Queue name required
164	Cannot open queue
165	Cannot open file
166	Cannot put to queue
167	No file name specified (Send)
168	Message length is too small to send data
169	Sending file has changed
170	Cannot get from queue
171	Cannot write to file
172	CorrelId is invalid
173	MsgId is invalid
174	No messages to receive
175	File for delete is not unique

Examples

This command lists all files and messages on the queue, `MY.QUEUE`, located on the default queue manager:

```
mqftrcvc -q MY.QUEUE -a
```

This command gets the first complete file on the queue, `MY.QUEUE`, located on queue manager `QM1`:

```
mqftrcvc -q MY.QUEUE -m QM1 -g
```

This command gets the complete file, named `My document.txt`, on the queue, `MY.QUEUE`, located on the default queue manager:

```
mqftrcvc -q MY.QUEUE -g -f "My document.txt"
```

This command gets the complete file, named `My document.txt`, also marked `URGENT`, on the queue, `MY.QUEUE`, located on queue manager `QM1` :

```
mqftrcvc -q MY.QUEUE -m QM1 -g -f "My document.txt" -s "URGENT"
```

Related commands

mqftapp	Run File Transfer Application
mqftrcv	Receive file on server
mqftsnd	Send file from server
mqftsndc	Send file from client

Parent topic: The control commands

This build: January 26, 2011 11:55:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15840_

5.2.29. mqftsnd

»Send a file from a server using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).«

Purpose

The mqftsnd command is available with the File Transfer Application on WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform) servers only.

Use the mqftsnd command to send a file from a WebSphere MQ server using the File Transfer Application.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '3? -t *TargetQMgrName*\n' + '4? -v \n' + '5? -l *MsgLength*\n' + '6! -p yes\n' + '6?\n' + '6 -p no\n' + '6 -p queue\n' + '7? -s *UserData*\n' + '8 -f *FileName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-mqftsnd-- -q --QueueName--+----------------+----------------->\n'
+ '                              \'- -m --QMgrName-\'    \n'
+ '\n'
+ '>--+----------------------+--+------+--+-----------------+------>\n'
+ '    \'- -t --TargetQMgrName-\'  \'- -v -\'  \'- -l --MsgLength-\'    \n'
+ '\n'
+ '    .- -p  yes---.                                          \n'
+ '>--+------------+--+----------------+-- -f --FileName---------><\n'
+ '    +- -p  no----+  \'- -s --UserData-\'                     \n'
+ '    \'- -p  queue-\'                                          \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-q *QueueName*

The local name of the destination queue.

-f *FileName*

The name of the file to be transmitted. The fully qualified file name can be specified. If the file name contains embedded spaces, it must be specified in double quotes. One file can be specified only, and you cannot use wildcard characters.

Note: The file is not deleted from its original location during a send.

Optional parameters

-m *QMgrName*

The name of the queue manager that has access to the file at its origin. If you omit this parameter, the default queue manager is used.

-t *TargetQMgrName*

The name of the queue manager that hosts the destination queue. If you omit this parameter, the queue manager specified by *QMgrName* is used.

-v

Return the *CorrelId* of the file.

-l *MessageSize*

The maximum size of a segmented message in bytes.

If a file is too large to be sent as a single message, the file is segmented into a number smaller messages, known as segments, and all these segments are transmitted instead. When all the segments reach their destination, the target queue manager reassembles them to form the original file.

Specify a value between 250 and the queue manager's maximum message length. To determine the maximum message length, use the MQIA_MAX_MSG_LENGTH selector with the MQINQ call.

The default value is 100000.

-p *yes*

Messages are persistent. This is the default value.

-p *no*

Messages are not persistent.

-p *queue*

Messages persistence is defined by the queue.

-s ***UserData***

An character string that contains user information relevant to the file being sent. The content of this data is of no significance to the target queue manager.

Return codes

0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
69	Storage not available
71	Unexpected error
163	Queue name required
164	Cannot open queue
165	Cannot open file
166	Cannot put to queue
167	No file name specified (Send)
168	Message length is too small to send data
169	Sending file has changed
170	Cannot get from queue
171	Cannot write to file
172	CorrelId is invalid
173	MsgId is invalid
174	No messages to receive
175	File for delete is not unique

Examples

This command sends a file from the default queue manager, to the queue DEST.Q, located on queue manager QM2:

```
mqftsnd -q DEST.Q -t QM2 -f "My document.txt"
```

This command sends a file as nonpersistent messages from queue manager QM1, to the queue DEST.Q, located on the default queue manager, setting the maximum segment size to 50000 bytes:

```
mqftsnd -q DEST.Q -m QM1 -l 50000 -p no -f "C:\My Downloads\My document.idd"
```

Related commands

mqftapp	Run File Transfer Application
mqftrcv	Receive file on server
mqftrcvc	Receive file on client
mqftsndc	Send file from client

Parent topic: The control commands

This build: January 26, 2011 11:55:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15850_

5.2.30. mqftsndc

Send a file from a client using the WebSphere MQ file transfer application (Windows and Linux x86 platforms only).

Purpose

The mqftsndc command is available with the File Transfer Application on WebSphere® MQ for Windows, and WebSphere MQ for Linux (x86 platform) clients only.

Use the mqftsndc command to send a file from a WebSphere MQ client using the File Transfer Application.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '3? -t *TargetQMgrName*\n' + '4? -v \n' + '5? -l *MsgLength*\n' + '6! -p yes\n' + '6?\n' + '6 -p no\n' +

'6 -p queue\n' + '7? -s *UserData*\n' + '8 -f *FileName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-mqftsndc-- -q --QueueName--+----------------+----------------->\n'
+ '                               \'- -m --QMgrName-\'    \n'
+ '\n'
+ '>--+----------------------+--+------+--+-----------------+------>\n'
+ '    \'- -t --TargetQMgrName-\'  \'- -v -\'  \'- -l --MsgLength-\'    \n'
+ '\n'
+ '    .- -p  yes---.                                          \n'
+ '>--+------------+--+----------------+-- -f --FileName----------><\n'
+ '    +- -p  no----+  \'- -s --UserData-\'                     \n'
+ '    \'- -p  queue-\'                                          \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-q ***QueueName***

The local name of the destination queue.

-f ***FileName***

The name of the file to be transmitted. The fully qualified file name can be specified. If the file name contains embedded spaces, it must be specified in double quotes. One file can be specified only, and you cannot use wildcard characters.

Note: The file is not deleted from its original location during a send.

Optional parameters

-m ***QMgrName***

The name of the queue manager that has access to the file at its origin. If you omit this parameter, the default queue manager is used.

-t ***TargetQMgrName***

The name of the queue manager that hosts the destination queue. If you omit this parameter, the queue manager specified by *QMgrName* is used.

-v

Return the *CorrelId* of the file.

-l ***MessageSize***

The maximum size of a segmented message in bytes.

If a file is too large to be sent as a single message, the file is segmented into a number smaller messages, known as segments, and all these segments are transmitted instead. When all the segments reach their destination, the target queue manager reassembles them to form the original file.

Specify a value between 250 and the queue manager's maximum message length. To determine the maximum message length, use the MQIA_MAX_MSG_LENGTH selector with the MQINQ call.

The default value is 100000.

-p ***yes***

Messages are persistent. This is the default value.

-p ***no***

Messages are not persistent.

-p ***queue***

Messages persistence is defined by the queue.

-s ***UserData***

An character string that contains user information relevant to the file being sent. The content of this data is of no significance to the target queue manager.

Return codes

0	Successful operation
36	Invalid arguments supplied

40	Queue manager not available
69	Storage not available
71	Unexpected error
163	Queue name required
164	Cannot open queue
165	Cannot open file
166	Cannot put to queue
167	No file name specified (Send)
168	Message length is too small to send data
169	Sending file has changed
170	Cannot get from queue
171	Cannot write to file
172	CorrelId is invalid
173	MsgId is invalid
174	No messages to receive
175	File for delete is not unique

Examples

This command sends a file from the default queue manager, to the queue DEST.Q, located on queue manager QM2:

```
mqftsndc -q DEST.Q -t QM2 -f "My document.txt"
```

This command sends a non-persistent file from queue manager QM1, to the queue DEST.Q, located on the default queue manager, setting the maximum segment size to 50000 bytes:

```
mqftsndc -q DEST.Q -m QM1 -l 50000 -p no -f "C:\My Downloads\My document.idd"
```

Related commands

mqftapp	Run File Transfer Application
mqftrcv	Receive file on server
mqftrcvc	Receive file on client
mqftsnd	Send file from server

Parent topic: The control commands

This build: January 26, 2011 11:55:36

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15860_

5.2.31. rcdmqimg

Write the image of an object or group of objects to the log for media recovery.

Purpose

Use the rcdmqimg command to write an image of an object, or group of objects, to the log for use in media recovery. This command can only be used when using linear logging. See Linear logging for more information about linear logging. Use the associated command rcrmqobj to recreate the object from the image.

rcdmqimg must be run manually or from an automated task you have created. The command does not run automatically as it needs to be run in accordance with, and as determined by, each individual customer's usage of WebSphere MQ.

Running rcdmqimg moves the log sequence number (LSN) forwards and frees up old log files for archival or deletion.

When determining when and how often to run rcdmqimg, consider these factors:

Disk space

If disk space is limited, regular running of rcdmqimg will release log files for archive or deletion.

Impact on normal system performance

rcdmqimg activity can take a long time if the queues on the system are deep. At this time, other system usage is slower and disk utilization increases because data is being copied from the queue files to the logs. Because of

this, the ideal time to run rcdmqimg is when the queues are empty and the system is not being heavily used.

You use this command with an active queue manager. Further activity on the queue manager is logged so that, although the image becomes out of date, the log records reflect any changes to the object.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

-t ObjectType *GenericObjName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-rcdmqimg--+----------------+--+------+--+------+-------------->\n'
+ '               \'- -m --QMgrName-\'   \'- -z -\'   \'- -l -\'    \n'
+ '\n'
+ '>-- -t ObjectType--GenericObjName------------------------------><\n'
+ '\n'
+ '
```

');} //]]>

Required parameters

GenericObjName

The name of the object to record. This parameter can have a trailing asterisk to record that any objects with names matching the portion of the name before the asterisk.

This parameter is required *unless* you are recording a queue manager object or the channel synchronization file. Any object name you specify for the channel synchronization file is ignored.

-t ***ObjectType***

The types of object for which to record images. Valid object types are:

all and *	All the object types; **ALL** for objtype and * for GenericObjName
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	Channels
clntconn or **clcn**	Client connection channels
catalog or **ctlg**	An object catalog
listener or **lstr**	Listeners
namelist or **nl**	Namelists
process or **prcs**	Processes
queue or **q**	All types of queue
qalias or **qa**	Alias queues
qlocal or **ql**	Local queues
qmodel or **qm**	Model queues
qremote or **qr**	Remote queues
qmgr	Queue manager object
service or **srvc**	Service
syncfile	Channel synchronization file.
topic or **top**	Topics

Note: When using WebSphere® MQ for UNIX systems, you need to prevent the shell from interpreting the meaning of special characters, for example, *. How you do this depends on the shell you are using, but might involve the use of single quotation marks, double quotation marks, or a backslash.

Optional parameters

-m ***QMgrName***

The name of the queue manager for which to record images. If you omit this, the command operates on the default queue manager.

-z

Suppresses error messages.

-l

Writes messages containing the names of the oldest log files needed to restart the queue manager and to perform media recovery. The messages are written to the error log and the standard error destination. (If you specify both the -z and -l parameters, the messages are sent to the error log, but not to the standard error destination.)

When issuing a sequence of rcdmqimg commands, include the -l parameter only on the last command in the sequence, so that the log file information is gathered only once.

Return codes

0	Successful operation
26	Queue manager running as a standby instance.
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
68	Media recovery not supported
69	Storage not available
71	Unexpected error
72	Queue manager name error
119	User not authorized
128	No objects processed
131	Resource problem
132	Object damaged
135	Temporary object cannot be recorded

Examples

The following command records an image of the queue manager object `saturn.queue.manager` in the log.

```
rcdmqimg -t qmgr -m saturn.queue.manager
```

Related commands

rcrmqobj	Recreate a queue manager object

Parent topic: The control commands

This build: January 26, 2011 11:55:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15870_

5.2.32. rcrmqobj

Recreate an object, or group of objects, from their images contained in the log.

Purpose

Use this command to recreate an object, or group of objects, from their images contained in the log. This command can only be used when using linear logging. Use the associated command, rcdmqimg, to record the object images to the log.

Use this command on a running queue manager. All activity on the queue manager after the image was recorded is logged. To recreate an object, replay the log to recreate events that occurred after the object image was captured.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

-t ObjectType *GenericObjName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-rcrmqobj--+----------------+--+------+-- -t ObjectType------->\n'
+ '              \'- -m --QMgrName-\'   \'- -z -\'                     \n'
+ '\n'
+ '>--GenericObjName-------------------------------------------><\n'
+ '\n'
+ '
```

');} //]]>

Required parameters

GenericObjName

The name of the object to re-create. This parameter can have a trailing asterisk to re-create any objects with names matching the portion of the name before the asterisk.

This parameter is required *unless* the object type is the channel synchronization file; any object name supplied for this object type is ignored.

-t ObjectType

The types of object to re-create. Valid object types are:

* or **all**	All object types
authinfo	Authentication information object, for use with Secure Sockets Layer (SSL) channel security
channel or **chl**	Channels
clntconn or **clcn**	Client connection channels
clchltab	Client channel table
listener or **lstr**	Listener
namelist or **nl**	Namelists
process or **prcs**	Processes
queue or **q**	All types of queue
qalias or **qa**	Alias queues
qlocal or **ql**	Local queues
qmodel or **qm**	Model queues
qremote or **qr**	Remote queues
service or **srvc**	Service
syncfile	Channel synchronization file. You can use this option when circular logs are configured but `syncfile` will fail if the channel scratchpad files, which are used to rebuild `syncfile`, are damaged or missing.
topic or **top**	Topics

Note: When using WebSphere® MQ for UNIX systems, you need to prevent the shell from interpreting the meaning of special characters, for example, *. How you do this depends on the shell you are using, but might involve the use of single quotation marks, double quotation marks, or a backslash.

Optional parameters

-m *QMgrName*

The name of the queue manager for which to recreate objects. If omitted, the command operates on the default queue manager.

-z

Suppresses error messages.

Return codes

0	Successful operation
» 26 «	» Queue manager running as a standby instance. «
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
66	Media image not available
68	Media recovery not supported
69	Storage not available
71	Unexpected error
72	Queue manager name error
119	User not authorized
128	No objects processed
135	Temporary object cannot be recovered
136	Object in use

Examples

1. **The following command recreates all local queues for the default queue manager:**

```
rcrmqobj -t ql *
```

2. The following command recreates all remote queues associated with queue manager `store`:

```
rcrmqobj -m store -t qr *
```

Related commands

rcdmqimg	Record an object in the log

Parent topic: The control commands

This build: January 26, 2011 11:55:37

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15880_

5.2.33. rmvmqinf

Remove WebSphere MQ configuration information (»Windows and «Unix platforms only).

Purpose

Use the rmvmqinf command to remove WebSphere MQ configuration information.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

StanzaName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '               .- -s -QueueManager-.                  \n'
+ '>>-rmvmqinf -+-------------------+--StanzaName-----------------><\n'
+ '               \'- -s -StanzaType---\'                  \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

StanzaName

The name of the stanza. That is, the value of the key attribute that distinguishes between multiple stanzas of the same type.

Optional parameters

-s *StanzaType*

The type of stanza to remove. If omitted, a QueueManager stanza is removed.
The only supported value of *StanzaType* is `QueueManager`.

Return codes

0	Successful operation
5	Queue manager is running
26	Queue manager is running as a standby instance
28	An error occurred removing the queue manager from the Service Control Manager
39	Bad command line parameters
44	Stanza does not exist
49	Queue manager is stopping
69	Storage is not available
71	Unexpected error
72	Queue manager name error

Example

```
rmvmqinf QM.NAME
```

Usage notes

Use rmvmqinf to remove an instance of a » multi-instance « queue manager.

To use this command you must be a WebSphere MQ administrator and a member of the `mqm` group.

Related commands

addmqinf	Add queue manager configuration information
dspmqinf	Display queue manager configuration information

Parent topic: The control commands

This build: January 26, 2011 11:55:31

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15703_

5.2.34. rsvmqtrn

»Resolve in-doubt transactions«

Purpose

»Use the rsvmqtrn command to commit or back out internally or externally coordinated in-doubt transactions.

Note: Use this command only when you are certain that transactions cannot be resolved by the normal protocols. Issuing this command might result in the loss of transactional integrity between resource managers for a distributed transaction.

«

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

Transaction\n' + '3 -m *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-rsvmqtrn--+- -a ------------------------+-- -m --QMgrName--><\n'
+ '               \'-+- -b --------+--Transaction-\'                    \n'
+ '                  +- -c --------+                                    \n'
+ '                  \'- -r --RMID -\'                                    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-m *QMgrName*

The name of the queue manager.

Optional parameters

-a

The queue manager resolves all internally-coordinated, in-doubt transactions (that is, all global units of work).

-b

Backs out the named transaction. This flag is valid for externally-coordinated transactions (that is, for external units of work) only.

-c

Commits the named transaction. This flag is valid for externally-coordinated transactions (that is, external units of work) only.

-r *RMID*

The resource manager whose participation in the in-doubt transaction can be ignored. This flag is valid for internally-coordinated transactions only, and for resource managers that have had their resource manager configuration entries removed from the queue manager configuration information.

Note: The queue manager does not call the resource manager. Instead, it marks the resource manager's participation in the transaction as being complete.

Transaction

The transaction number of the transaction being committed or backed out. Use the dspmqtrn command to find the relevant transaction number. This parameter is required with the -b, -c, and -r *RMID* parameters.

Return codes

0	Successful operation
26	Queue manager running as a standby instance.
32	Transactions could not be resolved
34	Resource manager not recognized
35	Resource manager not permanently unavailable
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
85	Transactions not known

Related commands

dspmqtrn	Display list of prepared transactions

Parent topic: The control commands

This build: January 26, 2011 11:55:37

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15890_

5.2.35. runmqchi

Run a channel initiator process to automate starting channels.

Purpose

Use the runmqchi command to run a channel initiator process. For more information about the use of this command, refer to WebSphere MQ Intercommunication.

The channel initiator is started by default as part of the queue manager.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqchi--+-----------------------+--+----------------+----->\n'
+ '               \'- -q --InitiationQName-\'   \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-q *InitiationQName*

The name of the initiation queue to be processed by this channel initiator. If you omit it, SYSTEM.CHANNEL.INITQ is used.

-m *QMgrName*

The name of the queue manager on which the initiation queue exists. If you omit the name, the default queue

manager is used.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

If errors occur that result in return codes of either 10 or 20, review the queue manager error log that the channel is associated with for the error messages, and the system error log for records of problems that occur before the channel is associated with the queue manager. For more information about error logs, see Error logs.

Parent topic: The control commands

This build: January 26, 2011 11:55:37

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15900_

5.2.36. runmqchl

Start a sender or requester channel

Purpose

Use the runmqchl command to run either a sender (SDR) or a requester (RQSTR) channel.

The channel runs synchronously. To stop the channel, issue the MQSC command STOP CHANNEL.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqchl-- -c --ChannelName--+----------------+------------><\n'
+ '                                \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-c *ChannelName*

The name of the channel to run.

Optional parameters

-m *QMgrName*

The name of the queue manager with which this channel is associated. If you omit the name, the default queue manager is used.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

If return codes 10 or 20 are generated, review the error log of the associated queue manager for the error messages, and the system error log for records of problems that occur before the channel is associated with the queue manager.

Parent topic: The control commands

This build: January 26, 2011 11:55:37

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15910_

5.2.37. runmqdlq

Start the dead-letter queue handler to monitor and process messages on the dead-letter queue.

Purpose

Use the runmqdlq command to start the dead-letter queue (DLQ) handler, which monitors and handles messages on a dead-letter queue.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName \n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqdlq--+------------------------+---------------------><\n'
+ '               \'- QName --+------------+-\'    \n'
+ '                           \'- QMgrName -\'      \n'
+ '\n'
+ '
```

');} //]]>

Description

Use the dead-letter queue handler to perform various actions on selected messages by specifying a set of rules that can both select a message and define the action to be performed on that message.

The runmqdlq command takes its input from `stdin`. When the command is processed, the results and a summary are put into a report that is sent to `stdout`.

By taking `stdin` from the keyboard, you can enter runmqdlq rules interactively.

By redirecting the input from a file, you can apply a rules table to the specified queue. The rules table must contain at least one rule.

If you use the DLQ handler without redirecting stdin from a file (the rules table), the DLQ handler reads its input from the keyboard. In WebSphere® MQ for AIX®, Solaris, HP-UX, and Linux, the DLQ handler does not start to process the named queue until it receives an end_of_file (Ctrl+D) character. In WebSphere MQ for Windows, it does not start to process the named queue until you press the following sequence of keys: Ctrl+Z, Enter, Ctrl+Z, Enter.

For more information about rules tables and how to construct them, see The DLQ handler rules table.

Optional parameters

The MQSC command rules for comment lines and for joining lines also apply to the DLQ handler input parameters.

QName

The name of the queue to be processed.

If you omit the name, the dead-letter queue defined for the local queue manager is used. If you enter one or more blanks (' '), the dead-letter queue of the local queue manager is explicitly assigned.

QMgrName

The name of the queue manager that owns the queue to be processed.

If you omit the name, the default queue manager for the installation is used. If you enter one or more blanks (' '), the default queue manager for this installation is explicitly assigned.

Parent topic: The control commands

This build: January 26, 2011 11:55:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15920_

5.2.38. runmqdnm

Start processing messages on a queue using the .NET monitor (Windows only).

Purpose

The runmqdnm command applies to WebSphere® MQ for Windows only.

runmqdnm can be run from the command line, or as a triggered application.

Use the runmqdnm control command to start processing messages on an application queue with a .NET monitor.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

AssemblyName\n' + '2? -m *QMgrName*\n' + '3? -c *ClassName*\n' + '4? -u *UserParameter*\n' + '5? -s *Syncpoint*\n' + '6? -d *Conversion*\n' + '7? -n *MaxThreads*\n' + '8? -t *Timeout*\n' + '9? -b *BackoutThreshold*\n' + '10? -r *QueueName*\n' + '11? -p *ContextOption*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqdnm-- -q --QueueName-- -a --AssemblyName----------------->\n'
+ '\n'
+ '>--+----------------+--+-----------------+---------------------->\n'
+ '    \'- -m --QMgrName-\'   \'- -c --ClassName-\'    \n'
+ '\n'
+ '>--+---------------------+--+-----------------+------------------>\n'
+ '    \'- -u --UserParameter-\'   \'- -s --Syncpoint-\'    \n'
+ '\n'
+ '>--+-------------------+--+------------------+------------------->\n'
+ '    \'- -d --Conversion-\'   \'- -n --MaxThreads-\'    \n'
+ '\n'
+ '>--+---------------+--+----------------------+----------------->\n'
+ '    \'- -t --Timeout-\'   \'- -b --BackoutThreshold-\'    \n'
+ '\n'
+ '>--+-----------------+--+--------------------+---------------><\n'
+ '    \'- -r --QueueName-\'   \'- -p --ContextOption-\'    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-q *QueueName*

The name of the application queue to monitor.

-a *AssemblyName*

The name of the .NET assembly.

Optional parameters

-m *QMgrName*

The name of the queue manager that hosts the application queue.

If omitted, the default queue manager is used.

-c *ClassName*

The name of the .NET class that implements the IMQObjectTrigger interface. This class must reside in the specified assembly.

If omitted, the specified assembly is searched to identify classes that implement the IMQObjectTrigger interface:

- **If one class is found, then *ClassName* takes the name of this class.**
- **If no classes or multiple classes are found, then the .NET monitor is not started and a message is written to the console.**

-u *UserData*

User defined data. This data is passed to the Execute method when the .NET monitor calls it. User data must be comprised of ASCII characters only, with no double-quotes, NULLs, or carriage returns.

If omitted, null is passed to the Execute method.

-s *Syncpoint*

Specifies whether sync point control is required when messages are retrieved from the application queue.

Possible values are:

YES	Messages are retrieved under sync point control (MQGMO_SYNCPOINT).
NO	Messages are not retrieved under sync point control (MQGMO_NO_SYNCPOINT).
PERSISTENT	Persistent messages are retrieved under sync point control (MQGMO_SYNCPOINT_IF_PERSISTENT).

If omitted, the value of *Syncpoint* is dependent on your transactional model:

- **If distributed transaction coordination (DTC) is being used, then *Syncpoint* is specified as YES.**
- **If distributed transaction coordination (DTC) is not being used, then *Syncpoint* is specified as PERSISTENT.**

-d *Conversion*

Specifies whether data conversion is required when messages are retrieved from the application queue. Possible values are:

YES	Data conversion is required (MQGMO_CONVERT).
NO	Data conversion is not required (no get message option specified).

If omitted, *Conversion* is specified as NO.

-n *MaxThreads*

The maximum number of active worker threads.

If omitted, *MaxThreads* is specified as 20.

-t *Timeout*

The time, in seconds, that the .NET monitor will wait for further messages to arrive on the application queue. If you specify -1, the .NET monitor will wait indefinitely.

If omitted when run from the command line, the .NET monitor will wait indefinitely.

If omitted when run as a triggered application, the .NET monitor will wait for 10 seconds.

-b *BackoutThreshold*

Specifies the backout threshold for messages retrieved from the application queue. Possible values are:

-1	The backout threshold is taken from the application queue attribute, BOTHRESH.
0	The backout threshold is not set.
1 or more	Explicitly sets the backout threshold.

If omitted, *BackoutThreshold* is specified as -1.

-r *QueueName*

The queue to which messages, whose backout count exceeds the backout threshold, are put.

If omitted, the value of *QueueName* is dependent on the value of the BOQNAME attribute from the application queue:

- **If BOQNAME is non-blank, then *QueueName* takes the value of BOQNAME.**
- **If BOQNAME is blank, then *QueueName* is specified as the queue manager dead letter queue. If a dead letter queue has not been assigned to the queue manager, then backout processing is not available.**

-p *ContextOption*

Specifies whether context information from a message that is being backed out is passed to the backed out message. Possible values are:

NONE	No context information is passed.
IDENTITY	Identity context information is passed only.
ALL	All context information is passed.

If omitted, *ContextOption* is specified as ALL.

Return codes

0	Successful operation
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
71	Unexpected error

72	Queue manager name error
133	Unknown object name error

Parent topic: The control commands

This build: January 26, 2011 11:55:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15930_

5.2.39. runmqlsr

»Run a listener process to listen for remote requests on various communication protocols.«

Purpose

Use the runmqlsr command to start a listener process.

This command is run synchronously and will wait until the listener process has finished before returning to the caller.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
IPAddr\n' + '2.3? -b *Backlog*\n' + '2 lu62 -n *TpName*\n' + '2 netbios \n' + '2.1+ \n' + '2.1? \n' + '2.1 -a *Adapter*\n' + '2.1 -l *LocalName*\n' + '2.1 -e *Names*\n' + '2.1 -s *Sessions*\n' + '2.1 -o *Commands*\n' + '2 spx \n' + '2.1+ \n' + '2.1? \n' + '2.1 -x *Socket*\n' + '2.1 -b *Backlog*\n' + '3? -m *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqlsr-- -t ----------------------------------------------->\n'
+ '\n'
+ '>--+- tcp --+------------+--+--------------+--+---------------+-+-->\n'
+ '    |         \'- -p --Port-\'  \'- -i --IPAddr-\'  \'- -b --Backlog-\' |    \n'
+ '    +- lu62 -- -n --TpName---------------------------------------+   \n'
+ '    |             .--------------------.                           |   \n'
+ '    |             V                      |                           |   \n'
+ '    +- netbios ----+-----------------+-+-------------------------+   \n'
+ '    |               +- -a --Adapter---+                             |   \n'
+ '    |               +- -l --LocalName-+                             |   \n'
+ '    |               +- -e --Names-----+                             |   \n'
+ '    |               +- -s --Sessions--+                             |   \n'
+ '    |               \'- -o --Commands--\'                             |   \n'
+ '    |         .-------------------.                                 |   \n'
+ '    |         V                    |                                 |   \n'
+ '    \'- spx ----+---------------+-+-------------------------------\'   \n'
+ '                +- -x --Socket--+                                       \n'
+ '                \'- -b --Backlog-\'                                       \n'
+ '\n'
+ '>--+----------------+------------------------------------------><\n'
+ '    \'- -m --QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

-t

The transmission protocol to be used:

tcp	Transmission Control Protocol / Internet Protocol (TCP/IP)
lu62	SNA LU 6.2 (Windows only)
netbios	NetBIOS (Windows only)
spx	SPX (Windows only)

Optional parameters

-p *Port*

» The port number for TCP/IP. This flag is valid for TCP only. If you omit the port number, it is taken from the queue manager configuration information, or from defaults in the program. The default value is 1414. It must not exceed 65535. «

-i *IPAddr*

The IP address for the listener, specified in one of the following formats:

- IPv4 dotted decimal
- IPv6 hexadecimal notation
- Alphanumeric format

This flag is valid for TCP/IP only.

On systems that are both IPv4 and IPv6 capable you can split the traffic by running two separate listeners, one listening on all IPv4 addresses and one listening on all IPv6 addresses. If you omit this parameter, the listener listens on all configured IPv4 and IPv6 addresses.

-n *TpName*

The LU 6.2 transaction program name. This flag is valid only for the LU 6.2 transmission protocol. If you omit the name, it is taken from the queue manager configuration information.

-a *Adapter*

The adapter number on which NetBIOS listens. By default the listener uses adapter 0.

-l *LocalName*

The NetBIOS local name that the listener uses. The default is specified in the queue manager configuration information.

-e *Names*

The number of names that the listener can use. The default value is specified in the queue manager configuration information.

-s *Sessions*

The number of sessions that the listener can use. The default value is specified in the queue manager configuration information.

-o *Commands*

The number of commands that the listener can use. The default value is specified in the queue manager configuration information.

-x *Socket*

The SPX socket on which SPX listens. The default value is hexadecimal 5E86.

-m *QMgrName*

The name of the queue manager. By default the command operates on the default queue manager.

-b *Backlog*

The number of concurrent connection requests that the listener supports. See TCP, LU62, NETBIOS, and SPX for a list of default values and further information.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing: the AMQMSVRN process did not start.

Examples

The following command runs a listener on the default queue manager using the NetBIOS protocol. The listener can use a maximum of five names, five commands, and five sessions. These resources must be within the limits set in the queue manager configuration information.

```
runmqlsr -t netbios -e 5 -s 5 -o 5
```

Parent topic: The control commands

This build: January 26, 2011 11:55:38

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15940_

5.2.40. runmqsc

Run WebSphere MQ commands on a queue manager.

Purpose

Use the runmqsc command to issue MQSC commands to a queue manager. MQSC commands enable you to perform administration tasks, for example defining, altering, or deleting a local queue object. MQSC commands and their syntax are described in the WebSphere MQ Script (MQSC) Command Reference.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

DefaultQMgrName\n' + '4.2?\n' + '4.2 -m *LocalQMgrName*\n' + '5? *QMgrName*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqsc--+------+--+------+---------------------------------->\n'
+ '              \'- -e -\'   \'- -v -\'    \n'
+ '\n'
+ '>--+-------------------------------------------------+------->\n'
+ '    |                                 .- -m -DefaultQMgrName-. |    \n'
+ '    \'- -w --WaitTime--+------+--+----------------------+-\'    \n'
+ '                       \'- -x -\'   \'- -m -LocalQMgrName---\'      \n'
+ '\n'
+ '>--+----------+-------------------------------------------------><\n'
+ '    \'-QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Description

You can invoke the runmqsc command in three ways:

Verify command

Verify MQSC commands but do not run them. An output report is generated indicating the success or failure of each command. This mode is available on a local queue manager only.

Run command directly

Send MQSC commands directly to a local queue manager.

Run command indirectly

Run MQSC commands on a remote queue manager. These commands are put on the command queue on a remote queue manager and run in the order in which they were queued. Reports from the commands are returned to the local queue manager.

The runmqsc command takes its input from `stdin`. When the commands are processed, the results and a summary are put into a report that is sent to `stdout`.

By taking `stdin` from the keyboard, you can enter MQSC commands interactively.

By redirecting the input from a file, you can run a sequence of frequently-used commands contained in the file. You can also redirect the output report to a file.

Optional parameters

-e

Prevents source text for the MQSC commands from being copied into a report. This is useful when you enter commands interactively.

-m *LocalQMgrName*

The local queue manager that you want to use to submit commands to the remote queue manager. If you omit this parameter the local default queue manager is used to submit commands to the remote queue manager.

-v

Verifies the specified commands without performing the actions. This mode is only available locally. The -w and -x flags are ignored if they are specified at the same time.

-w *WaitTime*

Run the MQSC commands on another queue manager. You must have the required channel and transmission queues set up for this. See Preparing channels and transmission queues for remote administration for more information.

WaitTime

The time, in seconds, that runmqsc waits for replies. Any replies received after this are discarded, but the MQSC commands still run. Specify a time between 1 and 999 999 seconds.

Each command is sent as an Escape PCF to the command queue (SYSTEM.ADMIN.COMMAND.QUEUE) of the target queue manager.

The replies are received on queue SYSTEM.MQSC.REPLY.QUEUE and the outcome is added to the report. This can be defined as either a local queue or a model queue.

This flag is ignored if the -v flag is specified.

-x

The target queue manager is running under z/OS®. This flag applies only in indirect mode. The -w flag must also be specified. In indirect mode, the MQSC commands are written in a form suitable for the WebSphere® MQ for z/OS command queue.

QMgrName

The name of the target queue manager on which to run the MQSC commands, by default, the default queue manager.

Return codes

00	MQSC command file processed successfully
10	MQSC command file processed with errors; report contains reasons for failing commands
20	Error; MQSC command file not run

Examples

1. Enter this command at the command prompt:

   ```
   runmqsc
   ```

   Now you can enter MQSC commands directly at the command prompt. No queue manager name is specified, so the MQSC commands are processed on the default queue manager.

2. Use one of these commands, as appropriate in your environment, to specify that MQSC commands are to be verified only:

   ```
   runmqsc -v BANK < "/u/users/commfile.in"

   runmqsc -v BANK < "c:\users\commfile.in"
   ```

   This command verifies the MQSC commands in file `commfile.in`. The queue manager name is `BANK`. The output is displayed in the current window.

3. These commands run the MQSC command file `mqscfile.in` against the default queue manager.

   ```
   runmqsc < "/var/mqm/mqsc/mqscfile.in" > "/var/mqm/mqsc/mqscfile.out"

   runmqsc < "c:\Program Files\IBM\WebSphere MQ\mqsc\mqscfile.in" >
         "c:\Program Files\IBM\WebSphere MQ\mqsc\mqscfile.out"
   ```

   In this example, the output is directed to file `mqscfile.out`.

4. This command submits commands to the QMREMOTE queue manager, using QMLOCAL to submit the commands.

   ```
   runmqsc -w 30 -m QMLOCAL QMREMOTE
   ```

Parent topic: The control commands

This build: January 26, 2011 11:55:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15950_

5.2.41. runmqtmc

Start the trigger monitor on a client.

Purpose

Use the runmqtmc command to invoke a trigger monitor for a client. For further information about using trigger monitors, refer to the WebSphere MQ Application Programming Guide.

Once a trigger monitor has started, it will continuously monitor the specified initiation queue. The trigger monitor will not stop until the queue manager ends, see endmqm. While the client trigger monitor is running it keeps the

dead letter queue open.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

InitiationQName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqtmc--+----------------+--+----------------------+----->\n'
+ '               \'- -m --QMgrName-\'   \'- -q --InitiationQName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-m ***QMgrName***

The name of the queue manager on which the client trigger monitor operates, by default the default queue manager.

-q ***InitiationQName***

The name of the initiation queue to be processed, by default SYSTEM.DEFAULT.INITIATION.QUEUE.

Return codes

0	Not used. The client trigger monitor is designed to run continuously and therefore not to end. The value is reserved.
10	Client trigger monitor interrupted by an error.
20	Error; client trigger monitor not run.

Examples

For examples of using this command, refer to the WebSphere MQ Application Programming Guide.

Parent topic: The control commands

This build: January 26, 2011 11:55:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa15960_

5.2.42. runmqtrm

Start the trigger monitor on a server.

Purpose

Use the runmqtrm command to invoke a trigger monitor. For further information about using trigger monitors, refer to the WebSphere MQ Application Programming Guide.

Once a trigger monitor has started, it will continuously monitor the specified initiation queue. The trigger monitor will not stop until the queue manager ends, see endmqm. While the trigger monitor is running it keeps the dead letter queue open.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

InitiationQName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-runmqtrm--+----------------+--+----------------------+----->\n'
+ '               \'- -m --QMgrName-\'   \'- -q --InitiationQName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-m *QMgrName*

The name of the queue manager on which the trigger monitor operates, by default the default queue manager.

-q *InitiationQName*

Specifies the name of the initiation queue to be processed, by default SYSTEM.DEFAULT.INITIATION.QUEUE.

Return codes

0	Not used. The trigger monitor is designed to run continuously and therefore not to end. Hence a value of 0 would not be seen. The value is reserved.
10	Trigger monitor interrupted by an error.
20	Error; trigger monitor not run.

Parent topic: The control commands

This build: January 26, 2011 11:55:39

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15970_

5.2.43. setmqaut

Change the authorizations to a profile, object, or class of objects. Authorizations can be granted to, or revoked from, any number of principals or groups.

For more information about authorization service components, see Installable services, Service components, and Authorization service.

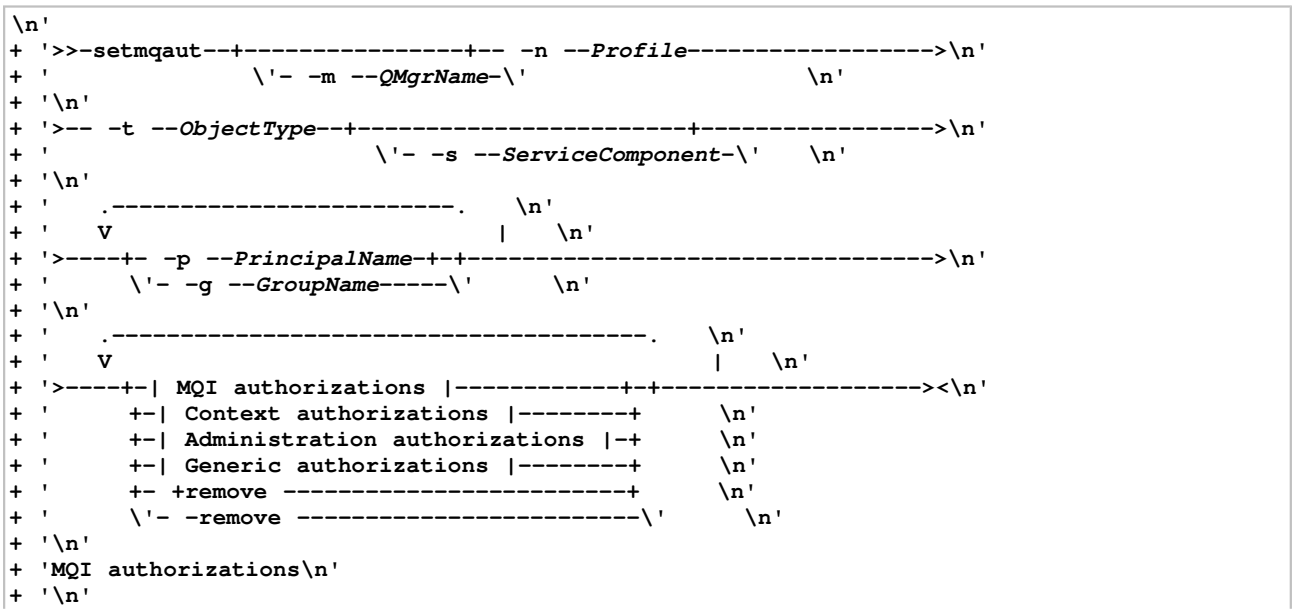
For more information about how authorizations work, see How authorizations work.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

**Profile -t *ObjectType*\n' + '4? -s *ServiceComponent*\n' + '5+ \n' + '5.0 -p *PrincipalName*\n' + '5.0 -g *GroupName*\n' + '6+ \n' + '6.0 %MQI authorizations\n' + '6.0 %Context authorizations\n' + '6.0 %Administration authorizations\n' + '6.0 %Generic authorizations\n' + '6.0 +remove \n' + '6.0 -remove \n' + '\n' + 'MQI authorizations\n' + '\n' + '1+ \n' + '1.0.0 +altusr \n' + '1.0.0 -altusr \n' + '1.0.0 +browse \n' + '1.0.0 -browse \n' + '1.0.0 +connect \n' + '1.0.0 -connect \n' + '1.0.0 +get \n' + '1.0.0 -get \n' + '1.0.0 +inq \n' + '1.0.0 -inq \n' + '1.0.0 +pub \n' + '1.0.0 -pub \n' + '1.0.0 +put \n' + '1.0.0 -put \n' + '1.0.0 +resume \n' + '1.0.0 -resume\n' + '1.0.0 +set \n' + '1.0.0 -set \n' + '1.0.0 +sub \n' + '1.0.0 -sub \n' + '\n' + 'Context authorizations\n' + '\n' + '1+ \n' + '1.0.0 +passall \n' + '1.0.0 -passall \n' + '1.0.0 +passid \n' + '1.0.0 -passid \n' + '1.0.0 +setall \n' + '1.0.0 -setall \n' + '1.0.0 +setid \n' + '1.0.0 -setid \n' + '\n' + 'Administration authorizations\n' + '\n' + '1+ \n' + '1.0.0 +chg \n' + '1.0.0 -chg \n' + '1.0.0 +clr \n' + '1.0.0 -clr \n' + '1.0.0 +crt \n' + '1.0.0 -crt \n' + '1.0.0 +dlt \n' + '1.0.0 -dlt \n' + '1.0.0 +dsp \n' + '1.0.0 -dsp \n' + '1.0.0 +ctrl \n' + '1.0.0 -ctrl \n' + '1.0.0 +ctrlx \n' + '1.0.0 -ctrlx \n' + '\n' + 'Generic authorizations\n' + '\n' + '1+ \n' + '1.0.0 +all \n' + '1.0.0 -all \n' + '1.0.0 +alladm \n' + '1.0.0 -alladm \n' + '1.0.0 +allmqi \n' + '1.0.0 -allmqi \n' + '1.0.0 +none \n' + '1.0.0 +system \n' + '1.0.0 -system \n' + '\n' + '

');break; default:document.write('



```
+ '    .--------------------.    \n'
+ '    V                     |    \n'
+ '|------+- +altusr --+---+-------------------------------------|\n'
+ '        +- -altusr --+        \n'
+ '        +- +browse --+        \n'
+ '        +- -browse --+        \n'
+ '        +- +connect -+        \n'
+ '        +- -connect -+        \n'
+ '        +- +get -----+        \n'
+ '        +- -get -----+        \n'
+ '        +- +inq -----+        \n'
+ '        +- -inq -----+        \n'
+ '        +- +pub -----+        \n'
+ '        +- -pub -----+        \n'
+ '        +- +put -----+        \n'
+ '        +- -put -----+        \n'
+ '        +- +resume --+        \n'
+ '        +- -resume---+        \n'
+ '        +- +set -----+        \n'
+ '        +- -set -----+        \n'
+ '        +- +sub -----+        \n'
+ '        \'- -sub -----\'        \n'
+ '\n'
+ 'Context authorizations\n'
+ '\n'
+ '    .--------------------.    \n'
+ '    V                     |    \n'
+ '|------+- +passall -+---+-------------------------------------|\n'
+ '        +- -passall -+        \n'
+ '        +- +passid --+        \n'
+ '        +- -passid --+        \n'
+ '        +- +setall --+        \n'
+ '        +- -setall --+        \n'
+ '        +- +setid ---+        \n'
+ '        \'- -setid ---\'        \n'
+ '\n'
+ 'Administration authorizations\n'
+ '\n'
+ '    .------------------.    \n'
+ '    V                   |    \n'
+ '|------+- +chg ---+---+-----------------------------------------|\n'
+ '        +- -chg ---+        \n'
+ '        +- +clr ---+        \n'
+ '        +- -clr ---+        \n'
+ '        +- +crt ---+        \n'
+ '        +- -crt ---+        \n'
+ '        +- +dlt ---+        \n'
+ '        +- -dlt ---+        \n'
+ '        +- +dsp ---+        \n'
+ '        +- -dsp ---+        \n'
+ '        +- +ctrl --+        \n'
+ '        +- -ctrl --+        \n'
+ '        +- +ctrlx -+        \n'
+ '        \'- -ctrlx -\'        \n'
+ '\n'
+ 'Generic authorizations\n'
+ '\n'
+ '    .-------------------.    \n'
+ '    V                    |    \n'
+ '|------+- +all ----+---+-------------------------------------|\n'
+ '        +- -all ----+        \n'
+ '        +- +alladm -+        \n'
+ '        +- -alladm -+        \n'
+ '        +- +allmqi -+        \n'
+ '        +- -allmqi -+        \n'
+ '        +- +none ---+        \n'
+ '        +- +system -+        \n'
+ '        \'- -system -\'        \n'
+ '\n'
+ '
');} //]]>
```

Description

Use setmqaut both to *grant* an authorization, that is, give a principal or user group permission to perform an operation, and to *revoke* an authorization, that is, remove the permission to perform an operation. You must specify the principals and user groups to which the authorizations apply, the queue manager, object type, and the profile name identifying the object or objects.

The authorizations that can be given are categorized as follows:

- **Authorizations for issuing MQI calls**
- **Authorizations for MQI context**
- **Authorizations for issuing commands for administration tasks**
- **Generic authorizations**

Each authorization to be changed is specified in an authorization list as part of the command. Each item in the list is a string prefixed by a plus sign (+) or a minus sign (-). For example, if you include +put in the authorization list, you grant authority to issue MQPUT calls against a queue. Alternatively, if you include -put in the authorization list, you revoke the authority to issue MQPUT calls.

You can specify any number of principals, user groups, and authorizations in a single command, but you must specify at least one principal or user group.

If a principal is a member of more than one user group, the principal effectively has the combined authorities of all those user groups. On Windows systems, the principal also has all the authorities that have been granted to it explicitly using the setmqaut command.

On UNIX systems, all authorities are held by user groups internally, not by principals. This has the following implications:

- **If you use the setmqaut command to grant an authority to a principal, the authority is actually granted to the primary user group of the principal. This means that the authority is effectively granted to all members of that user group.**
- **If you use the setmqaut command to revoke an authority from a principal, the authority is actually revoked from the primary user group of the principal. This means that the authority is effectively revoked from all members of that user group.**

To alter authorizations for a cluster sender channel that has been automatically generated by a repository, see WebSphere MQ Queue Manager Clusters. This book describes how the authority is inherited from a cluster receiver channel object.

Required parameters

-t *ObjectType*

The type of object for which to change authorizations.

Possible values are:

authinfo	An authentication information object
channel or **chl**	A channel
clntconn or **clcn**	A client connection channel
lstr or **listener**	A listener
namelist or **nl**	A namelist
process or **prcs**	A process
queue or **q**	A queue
qmgr	A queue manager
srvc or **service**	A service
topic or **top**	A topic

-n *Profile*

The name of the profile for which to change authorizations. The authorizations apply to all WebSphere® MQ objects with names that match the profile name specified. The profile name can be generic, using wildcard characters to specify a range of names as explained in Using OAM generic profiles on UNIX systems and Windows.

If you give an explicit profile name (without any wildcard characters), the object identified must exist.

This parameter is required, unless you are changing the authorizations of a queue manager, in which case you must *not* include it. To change the authorizations of a queue manager use the queue manager name, for example

```
setmqaut -m QMGR -t qmgr -p user1 +connect
```

where *QMGR* is the name of the queue manager and *user1* is the user requesting the change.

Optional parameters

-m *QMgrName*

The name of the queue manager of the object for which to change authorizations. The name can contain up to 48 characters.

This parameter is optional if you are changing the authorizations of your default queue manager.

-p *PrincipalName*

The name of the principal for which to change authorizations.

For WebSphere MQ for Windows only, the name of the principal can optionally include a domain name, specified in the following format:

```
userid@domain
```

For more information about including domain names on the name of a principal, see Principals and groups.

You must have at least one principal or group.

-g *GroupName*

The name of the user group for which to change authorizations. You can specify more than one group name, but each name must be prefixed by the -g flag. On Windows systems, you can use only local groups.

-s *ServiceComponent*

The name of the authorization service to which the authorizations apply (if your system supports installable authorization services). This parameter is optional; if you omit it, the authorization update is made to the first installable component for the service.

+remove or -remove

Remove all the authorities from WebSphere MQ objects that match the specified profile.

This option cannot be used with the option -t qmgr.

Authorizations

The authorizations to be granted or revoked. Each item in the list is prefixed by a plus sign (+), indicating that authority is to be granted, or a minus sign (-), indicating that authority is to be revoked.

For example, to grant authority to issue MQPUT calls, specify +put in the list. To revoke the authority to issue MQPUT calls, specify -put.

Table 1 shows the authorities that can be given to the different object types.

Table 1. Specifying authorities for different object types

Authority	Queue	Process	Queue manager	Namelist	Topic	Auth info	Clntconn	Channel	L
all	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
alladm[1]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
allmqi[2]	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No
none	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Ye
altusr	No	No	Yes	No	»No«	No	No	No	No
browse	Yes	No	No	No	No	No	No	No	No
chg	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
clr	Yes	No	No	No	Yes	No	No	No	No
connect	No	No	Yes	No	No	No	No	No	No
crt	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
ctrl	No	No	No	No	Yes	No	No	Yes	Ye
ctrlx	No	No	No	No	No	No	No	Yes	No
dlt	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
dsp	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Ye
get	Yes	No	No	No	No	No	No	No	No
pub	No	No	No	No	Yes	No	No	No	No
put	Yes	No	No	No	No	No	No	No	No
inq	Yes	Yes	Yes	Yes	No	Yes	No	No	No
passall	Yes	No	No	No	No	No	No	No	No
passid	Yes	No	No	No	No	No	No	No	No
resume	No	No	No	No	Yes	No	No	No	No
set	Yes	Yes	Yes	No	No	No	No	No	No
setall	Yes	No	Yes	No	No	No	No	No	No

setid	Yes	No	Yes	No	No	No	No	No	N
sub	No	No	No	No	Yes	No	No	No	N
»system«	»No«	»No«	»Yes«	»No«	No	»No«	»No«	»No«	»N

Note:

1. **alladm** authority is equivalent to the subset of the individual authorities chg, clr, crt, dlt, dsp, ctrl, and ctrlx appropriate to the o
2. **allmqi** authority is equivalent to the subset of the individual authorities altusr, browse, connect, get, inq, pub, put, resume, set to the object type.

Return codes

0	Successful operation
»26«	»Queue manager running as a standby instance.«
36	Invalid arguments supplied
40	Queue manager not available
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
133	Unknown object name
145	Unexpected object name
146	Object name missing
147	Object type missing
148	Invalid object type
149	Entity name missing
150	Authorization specification missing
151	Invalid authorization specification

Examples

1. **This example shows a command that specifies that the object on which authorizations are being given is the queue orange.queue on queue manager saturn.queue.manager. If the queue does not exist, the command fails.**

   ```
   setmqaut -m saturn.queue.manager -n orange.queue -t queue
            -g tango +inq +alladm
   ```

   The authorizations are given to a user group called tango, and the associated authorization list specifies that the user group can:

   - **Issue MQINQ calls**
   - **Perform all administration operations on that object**

2. **In this example, the authorization list specifies that a user group called foxy:**

   - **Cannot issue any MQI calls to the specified queue**
   - **Can perform all administration operations on the specified queue**

   If the queue does not exist, the command fails.

   ```
   setmqaut -m saturn.queue.manager -n orange.queue -t queue
            -g foxy -allmqi +alladm
   ```

3. **This example gives user1 full access to all queues with names beginning a.b. on queue manager qmgr1. The profile is persistent and applies to any object with a name that matches the profile.**

   ```
   setmqaut -m qmgr1 -n a.b.* -t q -p user1 +all
   ```

4. **This example deletes the specified profile.**

   ```
   setmqaut -m qmgr1 -n a.b.* -t q -p user1 -remove
   ```

5. **This example creates a profile with no authority.**

   ```
   setmqaut -m qmgr1 -n a.b.* -t q -p user1 +none
   ```

Related commands

dmpmqaut

Dump authority

dspmqaut

Display authority

Authorizations for MQI calls

Authorizations for context

Authorizations for commands

Authorizations for generic operations

Parent topic: The control commands

Related information
dmpmqaut (dump authority)
dspmqaut (display authority)

This build: January 26, 2011 11:55:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15980_

5.2.43.1. Authorizations for MQI calls

altusr	Use another user's authority for MQOPEN and MQPUT1 calls.
browse	Retrieve a message from a queue using an MQGET call with the BROWSE option.
connect	Connect the application to the specified queue manager using an MQCONN call.
get	Retrieve a message from a queue using an MQGET call.
inq	Make an inquiry on a specific queue using an MQINQ call.
pub	Publish a message on a topic using the MQPUT call.
put	Put a message on a specific queue using an MQPUT call.
resume	Resume a subscription using the MQSUB call.
set	Set attributes on a queue from the MQI using an MQSET call.
sub	Create, alter or resume a subscription to a topic using the MQSUB call.

Note: If you open a queue for multiple options, you must be authorized for each option.

Parent topic: setmqaut

This build: January 26, 2011 11:55:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15990_

5.2.43.2. Authorizations for context

passall	Pass all context on the specified queue. All the context fields are copied from the original request.
passid	Pass identity context on the specified queue. The identity context is the same as that of the request.
setall	Set all context on the specified queue. This is used by special system utilities.
setid	Set identity context on the specified queue. This is used by special system utilities.

Note: »To use setid or setall authority authorizations must be granted on both the appropriate queue object and also on the queue manager object. «

Parent topic: setmqaut

This build: January 26, 2011 11:55:40

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16000_

5.2.43.3. Authorizations for commands

chg	Change the attributes of the specified object.
clr	Clear the specified queue or a topic.
crt	Create objects of the specified type.
dlt	Delete the specified object.
dsp	Display the attributes of the specified object.
ctrl	For listeners and services, start and stop the specified channel, listener, or service. For channels, start, stop, and ping the specified channel. For topics, define, alter, or delete subscriptions.
ctrlx	Reset or resolve the specified channel.

Parent topic: setmqaut

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16010_

5.2.43.4. Authorizations for generic operations

all	Use all operations applicable to the object.
alladm	Use all administration operations applicable to the object.
allmqi	Use all MQI calls applicable to the object.
none	No authority. Use this authorization to create profiles without authority. When an authority is given to an object or group that was previously showing "none", then the authorization changes to the authority just applied. However, when the "none" authorization is added to a set object or group with an existing alternative authority, the authority does not change.
system	Use queue manager for internal system operations.

Parent topic: setmqaut

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16020_

5.2.44. setmqcrl

Administer CRL (certificate revocation list) LDAP definitions in an Active Directory (Windows only).

Purpose

The setmqcrl command applies to WebSphere® MQ for Windows only.

Use the setmqcrl command to configure and administer support for publishing CRL (certificate revocation list) LDAP definitions in an Active Directory.

A domain administrator must use this command, or setmqscp, initially to prepare the Active Directory for WebSphere MQ usage and to grant WebSphere MQ users and administrators the relevant authorities to access and update the WebSphere MQ Active Directory objects. You can also use the setmqcrl command to display all the currently configured CRL server definitions available on the Active Directory, that is, those definitions referred to by the queue manager's CRL namelist.

The only types of CRL servers supported are LDAP servers.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

```
QMgrName\n' + '4?  -d \n' + '\n' + '

');break; default:document.write('

\n'
+  '>>-setmqcrl--+-------------------------+---------------------->\n'
```

```
+ '               \'- -a --+----------------+-\'    \n'
+ '                        \'- -m --QMgrName-\'      \n'
+ '\n'
+ '>--+-------------------------+--+------+----------------------><\n'
+ '    \'- -r --+----------------+-\'  \'- -d -\'    \n'
+ '             \'- -m --QMgrName-\'                 \n'
+ '\n'
+ '
');} //]]>
```

Optional parameters

You must specify one of `-a` (add), `-r` (remove) or `-d` (display).

-a

Adds the WebSphere MQ client connections Active Directory container, if it does not already exist. You must be a user with the appropriate privileges to create subcontainers in the *System* container of your domain. The WebSphere MQ folder is called `CN=IBM-MQClientConnections`. Do not delete this folder in any other way than by using the setmqscp command.

-d

Displays the WebSphere MQ CRL server definitions.

-r

Removes the WebSphere MQ CRL server definitions.

-m [* | qmgr]

Modifies the specified parameter (`-a` or `-r`) so that only the specified queue manager is affected. You must include this option with the `-a` parameter.

*** | qmgr**

*** specifies that all queue managers are affected. This enables you to migrate a specific WebSphere MQ CRL server definitions file from one queue manager alone.**

Examples

The following command creates the IBM-MQClientConnections folder and allocates the required permissions to WebSphere MQ administrators for the folder, and to child objects created subsequently. (In this, it is functionally equivalent to `setmqscp -a`.)

```
setmqcrl -a
```

The following command migrates existing CRL server definitions from a local queue manager, Paint.queue.manager, to the Active Directory, deleting any other CRL definitions from the Active Directory first:

```
setmqcrl -a -m Paint.queue.manager
```

Parent topic: The control commands

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16040_

5.2.45. setmqprd

Enroll a WebSphere® MQ production license.

A license is normally enrolled as part of the installation process. For further information about the installation process, see the WebSphere MQ *Quick Beginnings* guide for your operating system.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal
');break; default:document.write('

```
\n'
+ '>>-setmqprd--LicenseFile---------------------------------------><\n'
+ '\n'
+ '
');} //]]>
```

Required parameters

LicenseFile

Specifies the fully-qualified name of the production license certificate file.

This is usually amqpcert.lic.

Parent topic: The control commands

This build: January 26, 2011 11:55:41

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16050_

5.2.46. setmqscp

Publish client connection channel definitions in an Active Directory (Windows only).

Purpose

The setmqscp command applies to WebSphere® MQ for Windows only.

Use the setmqscp command to configure and administer support for publishing client connection channel definitions in an Active Directory.

Initially, this command is used by a domain administrator to:

- **Prepare the Active Directory for WebSphere MQ use**
- **Grant WebSphere MQ users and administrators the relevant authorities to access and update the WebSphere MQ Active Directory objects**

You can also use the setmqscp command to display all the currently configured client connection channel definitions available on the Active Directory.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '4? -d \n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-setmqscp--+-------------------------+----------------------->\n'
+ '              \'- -a --+----------------+-\'    \n'
+ '                       \'- -m --QMgrName-\'      \n'
+ '\n'
+ '>--+---------------------------+--+------+----------------------><\n'
+ '    \'- -r --+----------------+-\'  \'- -d -\'    \n'
+ '             \'- -m --QMgrName-\'                \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

You must specify one of -a (add), -r (remove) or -d (display).

-a

Adds the WebSphere MQ client connections Active Directory container, if it does not already exist. You must be a user with the appropriate privileges to create subcontainers in the *System* container of your domain. The WebSphere MQ folder is called `CN=IBM-MQClientConnections`. Do not delete this folder in any other way than by using the `setmqscp -r` command.

-d

Displays the service connection points.

-r

Removes the service connection points. If you omit -m, and no client connection definitions exist in the IBM-MQClientConnections folder, the folder itself is removed from the Active Directory.

-m [* | qmgr]

Modifies the specified parameter (-a or -r) so that only the specified queue manager is affected.

*** | qmgr**

*** specifies that all queue managers are affected. This enables you to migrate a specific client connection table**

file from one queue manager alone, if required.

Examples

The following command creates the IBM-MQClientConnections folder and allocates the required permissions to WebSphere MQ administrators for the folder, and to child objects created subsequently:

```
setmqscp -a
```

The following command migrates existing client connection definitions from a local queue manager, Paint.queue.manager, to the Active Directory:

```
setmqscp -a -m Paint.queue.manager
```

The following command migrates all client connection definitions on the local server to the Active Directory:

```
setmqscp -a -m *
```

Parent topic: The control commands

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16060_

5.2.47. strmqcfg

Start the WebSphere® MQ Explorer (Windows and Linux x86 platforms only).

Purpose

The strmqcfg command is available on WebSphere MQ for Windows, and WebSphere MQ for Linux (x86 platform) only.

For WebSphere MQ for Windows only, note that if you use runas to execute this command, you must define the Environment Variable *APPDATA*.

On Linux, the WebSphere MQ Explorer might fail to start if you have more than one Eclipse installation. If this happens, start the WebSphere MQ Explorer using a different user ID to the one you use for the other Eclipse installation.

On Linux, to start the WebSphere MQ Explorer successfully, you must be able to write a file to your home directory, and the home directory must exist.

Syntax

The syntax of this command follows:
Syntax diagram format ○ Railroad diagram ○ Dotted decimal

\n' + '>>-strmqcfg--+------+--+------+--+------+----------><\n' + ' \'- -c -\' \'- -i -\' \'- -x -\' \n' + '\n' + '

');} //]]>

Optional parameters

`-c`

`-clean` is passed to Eclipse. This causes Eclipse to delete any cached data used by the Eclipse runtime.

`-i`

`-init` is passed to Eclipse. This causes Eclipse to discard configuration information used by the Eclipse runtime.

`-x`

Output debug messages to the console.

Parent topic: The control commands

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16070_

5.2.48. strmqcsv

Start the command server for a queue manager.

Purpose

Use the strmqcsv command to start the command server for the specified queue manager. This enables WebSphere® MQ to process commands sent to the command queue.

If the queue manager attribute, SCMDSERV, is specified as QMGR then changing the state of the command server using strmqcsv does not effect how the queue manager acts upon the SCMDSERV attribute at the next restart.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

');break; default:document.write('

```
\n'
+ '>>-strmqcsv--+-----+--+----------+--------------------------><\n'
+ '              \'- -a-\'   \'-QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Required parameters

None

Optional parameters

-a

Blocks the following PCF commands from modifying or displaying authority information:

- Inquire authority records (MQCMD_INQUIRE_AUTH_RECS)
- Inquire entity authority (MQCMD_INQUIRE_ENTITY_AUTH)
- Set authority record (MQCMD_SET_AUTH_REC).
- Delete authority record (MQCMD_DELETE_AUTH_REC).

QMgrName

The name of the queue manager on which to start the command server. If omitted, the default queue manager is used.

Return codes

0	Command completed normally
10	Command completed with unexpected results
20	An error occurred during processing

Examples

The following command starts a command server for queue manager `earth`:

```
strmqcsv earth
```

Related commands

endmqcsv	End a command server
dspmqcsv	Display the status of a command server

Parent topic: The control commands

This build: January 26, 2011 11:55:41

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16080_

5.2.49. strmqm

Start a queue manager or ready it for standby operation.

Purpose

»Use the strmqm command to start a queue manager.«

If the queue manager start up takes more than a few seconds WebSphere® MQ shows intermittent messages detailing the start up progress. For more information about these messages see WebSphere MQ Messages.

Syntax

Syntax diagram format ○ Railroad diagram ○ Dotted decimal

QMgrName\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-strmqm--+------+--+-----+--+-------+--+------------------+--->\n'
+ '            +- -c -+  \'- -f \'  +- -si -+  \'- -d--Information-\'    \n'
+ '            +- -r -+            \'- -ss -\'                          \n'
+ '            +- -a -+                                                \n'
+ '            \'- -x -\'                                                \n'
+ '\n'
+ '>--+------+--+------+--+----------+--------------------------><\n'
+ '    \'- -z -\'  \'- -ns-\'  \'-QMgrName-\'    \n'
+ '\n'
+ '
```

');} //]]>

Optional parameters

-a

Activate the specified backup queue manager. The backup queue manager is not started.

When activated, a backup queue manager can be started using the control command `strmqm` *QMgrName*. The requirement to activate a backup queue manager prevents accidental startup.

When activated, a backup queue manager can no longer be updated.

For more information on using backup queue managers, see Backing up and restoring WebSphere MQ.

-c

»Starts the queue manager, redefines the default and system objects, then stops the queue manager. Any existing system and default objects belonging to the queue manager are replaced if you specify this flag, and any non-default system object values will be reset (for example, the value of MCAUSER is set to blank).

Use the crtmqm command to create the default and system objects for a queue manager.

«

-d ***Information***

Specifies whether information messages are displayed. Possible values for *Information* follow:

all	All information messages are displayed. This is the default value.
minimal	The minimal number of information messages are displayed.
none	No information messages are displayed. This parameter is equivalent to **-z**.

The *-z* parameter takes precedence over this parameter.

»-f«

»Use this option if you *know* a queue manager is not starting because its data directories are missing or corrupted.

The strmqm `-f` *qmname* command attempts to recreate the queue manager data directory and reset file permissions. If it is successful, the queue manager starts, unless the queue manager configuration information is missing. If the queue manager fails to start because the configuration information is missing, recreate the configuration information, and restart the queue manager.

In releases of WebSphere MQ earlier than 7.0.1, strmqm, with no `-f` option, automatically repaired missing data directories and then tried to start. This behavior has changed.

From WebSphere MQ Version 7.0.1 onwards, the default behavior of strmqm, with no `-f` option, is *not* to recover missing or corrupt data directories automatically, but to report an error, such as AMQ6235 or AMQ7001, and *not* start the queue manager.

You can think of the `-f` option as performing the recover actions that used to be performed automatically by strmqm.

The reason for the change to the behavior of strmqm is that with the support for networked file storage in WebSphere MQ Version 7.0.1, the most likely cause of missing or corrupt queue manager data directories is a configuration error that can be rectified, rather than the data directories are corrupt or irretrievably unavailable.

You should *not* use strmqm `-f` to recreate the queue manager data directories if you can restore the directories by correcting the configuration.

Possible solutions to problems with strmqm are to make the networked file storage location accessible to the queue manager, or to ensure the gid and uid of the mqm group and userid on the server hosting the queue manager match the gid and uid of the mqm group and userid on the server hosting the queue manager data directory.

From WebSphere MQ Version 7.0.1, if you are performing media recovery for a queue manager, then you should use the `-f` option to recreate the queue manager data directory.

-ns

Prevents any of the following processes from starting automatically when the queue manager starts:

- The channel initiator
- The command server
- Listeners
- Services

-r

Updates the backup queue manager. The backup queue manager is not started.

WebSphere MQ updates the backup queue manager's objects by reading the queue manager log and replaying updates to the object files.

For more information on using backup queue managers, see Backing up and restoring WebSphere MQ.

-si

Interactive (manual) queue manager startup type. This option is available on WebSphere MQ for Windows only.

The queue manager runs under the logged on (interactive) user. Queue managers configured with interactive startup end when the user who started them logs off.

If you set this parameter, it overrides any startup type set previously by the crtmqm command, the amqmdain command, or the WebSphere MQ Explorer.

If you do not specify a startup type of either `-si` or `-ss`, the queue manager startup type specified on the crtmqm command is used.

-ss

Service (manual) queue manager startup type. This option is available on WebSphere MQ for Windows only.

The queue manager runs as a service. Queue managers configured with service startup continue to run even after the interactive user has logged off.

If you set this parameter, it overrides any startup type set previously by the crtmqm command, the amqmdain command, or the WebSphere MQ Explorer.

-x

Start an instance of a multi-instance queue manager on the local server, permitting it to be highly available. If an instance of the queue manager is not already running elsewhere, the queue manager starts and the instance becomes active. The active instance is ready to accept local and remote connections to the queue manager on the local server.

If a multi-instance queue manager instance is already active on a *different* server the new instance becomes a standby, permitting it to takeover from the active queue manager instance. While it is in standby, it cannot accept local or remote connections.

You may not start a second instance of a queue manager on the *same* server.

The default behavior, omitting the -x optional parameter, is to start the instance as a single instance queue manager, forbidding standby instances from being started.

-z

Suppresses error messages.

This flag is used within WebSphere MQ to suppress unwanted information messages. Because using this flag could result in loss of information, do not use it when entering commands on a command line.

This parameter takes precedence over the -d parameter.

QMgrName

The name of a local queue manager. If omitted, the default queue manager is used.

Return codes

0	Queue manager started
3	Queue manager being created
5	Queue manager running
16	Queue manager does not exist
23	Log not available
24	A process that was using the previous instance of the queue manager has not yet disconnected.
30	A standby instance of the queue manager started. The active instance is running elsewhere
31	The queue manager already has an active instance. The queue manager permits standby instances.
39	Invalid parameter specified
43	The queue manager already has an active instance. The queue manager does not permit standby instances.
47	The queue manager already has the maximum number of standby instances
49	Queue manager stopping
69	Storage not available
71	Unexpected error
72	Queue manager name error
74	The WebSphere MQ service is not started.
100	Log location invalid
119	User not authorized to start the queue manager

Examples

The following command starts the queue manager `account`:

```
strmqm account
```

Related commands

crtmqm	Create a queue manager
dltmqm	Delete a queue manager
endmqm	End a queue manager

Parent topic: The control commands

This build: January 26, 2011 11:55:42

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16090_

5.2.50. strmqtrc

Enable trace at a specified level of detail, or report the level of tracing in effect.

Purpose

Use the strmqtrc command to enable tracing.

Syntax

The syntax of this command is as follows:
Syntax diagram format ○ Railroad diagram ○ Dotted decimal
TraceType\n' + '5? -x *TraceType*\n' + '6? -l *MaxSize*\n' + '7? -d\n' + '7.1 0\n' + '7.1 -1\n' + '7.1 *NumOfBytes*\n' + '8? -

i *PidTids*\n' + '9? -p *Apps*\n' + '10? -s \n' + '11? -b *StartTrigger*\n' + '12? -c *StopTrigger*\n' + '\n' + '

');break; default:document.write('

```
\n'
+ '>>-strmqtrc--+----------------+--+------+----------------------->\n'
+ '              \'- -m --QMgrName-\'  \'- -e -\'    \n'
+ '\n'
+ '    .---------------------.                          \n'
+ '    V                     |                            \n'
+ '>----+-----------------+-+--+----------------+-----------------><\n'
+ '      \'- -t --TraceType-\'     \'- -x --TraceType-\'    \n'
+ '\n'
+ '>--+---------------+--+--------------------+-----------------><\n'
+ '    \'- -l --MaxSize-\'  \'- -d--+- 0----------+-\'    \n'
+ '                                 +- -1---------+      \n'
+ '                                 \'- NumOfBytes-\'      \n'
+ '\n'
+ '>--+---------------+--+-----------+--+------+-----------------><\n'
+ '    \'- -i --PidTids-\'  \'- -p --Apps-\'  \'- -s -\'    \n'
+ '\n'
+ '>--+--------------------+--+------------------+--------------><\n'
+ '    \'- -b --StartTrigger-\'  \'- -c --StopTrigger-\'    \n'
+ '\n'
+ '
```

');} //]]>

Description

The strmqtrc command enables tracing. The command has optional parameters that specify the level of tracing you want:

- One or more queue managers
- Levels of trace detail
- One or more Websphere MQ processes. The processes can be either part of the Websphere MQ product or customer applications that use the Websphere MQ API
- Specific threads within customer applications, either by WebSphere MQ thread number or by operating system thread number
- Events. These can be either the entry or exit from internal WebSphere MQ functions or the occurrence of a first failure data capture (FDC).

Each combination of parameters on an individual invocation of the command are interpreted by WebSphere MQ as having a logical AND between them. You can invoke the strmqtrc command multiple times, regardless of whether tracing is already enabled. If tracing is already enabled, the trace options that are in effect are modified to those specified on the most recent invocation of the command. Multiple invocations of the command, without an intervening enqmqtrc command, are interpreted by WebSphere MQ as having a logical OR between them. The maximum number of concurrent strmqtrc commands that can be in effect at one time is 16.

In WebSphere® MQ for Windows, the output file is created in the `\<mqmwork>\trace` directory, where `<mqmwork>` is the directory selected to hold WebSphere MQ data files.

In WebSphere MQ for AIX®, HP-UX, Solaris, and Linux, the output file is created in the directory `/var/mqm/trace`.

For examples of trace data generated by this command see Tracing.

Optional parameters

-m *QMgrName*

The name of the queue manager to trace.

The following wildcards are allowed: asterisk (*), replacing zero or more characters, and question mark (?), replacing any single character. In command environments such as the UNIX shell, where the asterisk (*) and question mark (?) characters have special meaning, you must either escape the wildcard character or enclose it in quotation marks to prevent the command environment from operating on the wildcard character.

-e

Requests early tracing of all processes, making it possible to trace the creation or startup of a queue manager. If you include this flag, any process belonging to any component of any queue manager traces its early processing. The default is not to perform early tracing.

You cannot use the -e flag with the -m flag, -i flag, the -p flag or the -b flag. If you try to use the -e flag with the -i flag, the -p flag or the -b flag, then an error message is issued.

-t *TraceType*

The points to trace and the amount of trace detail to record. By default all trace points are enabled and a default-detail trace is generated.

Alternatively, you can supply one or more of the options in the following list. For each *tracetype* value you specify, including `-t all`, specify either `-t parms` or `-t detail` to obtain the appropriate level of trace detail. If you do not specify either `-t parms` or `-t detail` for any particular trace type, only a default-detail trace is generated for that trace type.

If you supply multiple trace types, each must have its own -t flag. You can include any number of -t flags, provided that each has a valid trace type associated with it.

It is not an error to specify the same trace type on multiple -t flags.

all	Output data for every trace point in the system (the default). The all parameter activates tracing at default detail level.
api	Output data for trace points associated with the MQI and major queue manager components.
commentary	Output data for trace points associated with comments in the WebSphere MQ components.
comms	Output data for trace points associated with data flowing over communications networks.
csdata	Output data for trace points associated with internal data buffers in common services.
csflows	Output data for trace points associated with processing flow in common services.
detail	Activate tracing at high-detail level for flow processing trace points.
Explorer	Output data for trace points associated with the WebSphere MQ Explorer.
java	Output data for trace points associated with applications using the WebSphere MQ classes for Java API.
lqmdata	Output data for trace points associated with internal data buffers in the local queue manager.
lqmflows	Output data for trace points associated with processing flow in the local queue manager.
otherdata	Output data for trace points associated with internal data buffers in other components.
otherflows	Output data for trace points associated with processing flow in other components.
parms	Activate tracing at default-detail level for flow processing trace points.
remotedata	Output data for trace points associated with internal data buffers in the communications component.
remoteflows	Output data for trace points associated with processing flow in the communications component.
servicedata	Output data for trace points associated with internal data buffers in the service component.
serviceflows	Output data for trace points associated with processing flow in the service component.
soap	Output data for trace points associated with WebSphere MQ Transport for SOAP.
ssl	Output data associated with using GSKit to enable Secure Sockets Layer (SSL) channel security.
versiondata	Output data for trace points associated with the version of WebSphere MQ running.

-x *TraceType*

The points not to trace. By default all trace points are enabled and a default-detail trace is generated. The trace points you can specify are those listed for the -t flag.

You can use the `-x` flag with *tracetype* values to exclude those entry points you do not want to record. This is useful in reducing the amount of trace produced.

If you supply multiple trace types, each must have its own -x flag. You can include any number of -x flags, provided that each has a valid *tracetype* associated with it.

-l *MaxSize*

The maximum size of a trace file (`AMQ`*ppppp*`.`*qq*`.TRC`) in megabytes (MB). For example, if you specify a MaxSize of 1, the size of the trace is limited to 1 MB.

When a trace file reaches the specified maximum, it is renamed to `AMQ`*ppppp*`.`*qq*`.TRS` and a new `AMQ`*ppppp*`.`*qq*`.TRC` file is started. If a previous copy of an `AMQ`*ppppp*`.`*qq*`.TRS` file exists, it is deleted.

The highest value that *MaxSize* can be set to is 2048 MB.

-d `0`

Trace no user data.

-d -1 or all

Trace all user data.

-d *NumOfBytes*

- **For a communication trace; trace the specified number of bytes of data including the transmission segment header (TSH).**
- **For an MQPUT or MQGET call; trace the specified number of bytes of message data held in the message buffer.**
- **Values between one and 15 are not allowed.**

-i *PidTids*

Process identifier (PID) and thread identifier (TID) to which the trace generation is restricted. You cannot use the -i flag with the -e flag. If you try to use the -i flag with the -e flag, then an error message is issued. This parameter should be used only under the guidance of IBM® Service personnel.

»This parameter is not supported for .NET clients if NMQ_MQ_LIB is set to `managed`, so that the client uses managed WebSphere MQ problem diagnostics.«

-p *Apps*

The named processes to which the trace generation is restricted. *Apps* is a comma-separated list. You must specify each name in the list exactly as the program name would be displayed in the "Program Name" FDC header. Asterisk (*) or question mark (?) wildcards are allowed. You cannot use the -p flag with the -e flag. If you try to use the -p flag with the -e flag, then an error message is issued.

»This parameter is not supported for .NET clients if NMQ_MQ_LIB is set to `managed`, so that the client uses managed WebSphere MQ problem diagnostics.«

-s

Reports the tracing options that are currently in effect. You must use this parameter on its own with no other parameters.

A limited number of slots are available for storing trace commands. When all slots are in use, then no more trace commands can be accepted unless they replace an existing slot. Slot numbers are not fixed, so if the command in slot number 0 is removed, for example by an endmqtrc command, then all the other slots move up, with slot 1 becoming slot 0, for example. An asterisk (*) in a field means that no value is defined, and is equivalent to the asterisk wildcard.

An example of the output from this command is as follows:

```
Listing Trace Control Array

Used slots = 2 of 15

EarlyTrace     [OFF]
TimedTrace     [OFF]
TraceUserData [0]
MaxSize         [0]
Trace Type     [1]

Slot position 1

Untriggered
Queue Manager [avocet]
Application    [*]
PID.TID         [?]
TraceOptions   [1f4ffff]
TraceInterval     [0]
Trace Start Time [0]
Trace Stop  Time [0]
Start Trigger [KN346050K]
Start Trigger [KN346080]

Slot position 2

Untriggered
Queue Manager [*]
Application    [*]
PID.TID         [?]
TraceOptions   [1fcffff]
TraceInterval     [0]
Trace Start Time [0]
Trace Stop  Time [0]
Start Trigger [KN346050K]
```

```
Start Trigger [KN346080]
```

»This parameter is not supported for .NET clients if NMQ_MQ_LIB is set to `managed`, so that the client uses managed WebSphere MQ problem diagnostics.«

-b *Start_Trigger*

FDC probe IDs for which tracing should be turned on. *Start_Trigger* is a comma-separated list of FDC probe IDs. You can use asterisk (*) and question mark (?) wildcards in the specification of probe IDs. You cannot use the -b flag with the -e flag. If you try to use the -b flag with the -e flag, then an error message is issued. This parameter should be used only under the guidance of IBM Service personnel.

Table 1.

Start_Trigger	***Effect***
FDC=comma-separated list of FDC probe IDs.	Turns tracing on when any FDCs with the specified FDC probe IDs are generated.

»This parameter is not supported for .NET clients if NMQ_MQ_LIB is set to `managed`, so that the client uses managed WebSphere MQ problem diagnostics.«

-c *Stop_Trigger*

FDC probe IDs for which tracing should be turned off, or interval in seconds after which tracing should be turned off. *Stop_Trigger* is a comma-separated list of FDC probe IDs. You can use asterisk (*) and question mark (?) wildcards in the specification of probe IDs. This parameter should be used only under the guidance of IBM Service personnel.

Table 2.

Stop_Trigger	***Effect***
FDC=comma-separated list of FDC probe IDs.	Turns tracing off when any FDCs with the specified FDC probe IDs are generated.
interval=n where n is an unsigned integer between 1 and 32,000,000.	Turns tracing off n seconds after it starts or, if it tracing is already enabled, turns tracing off n seconds after this instance of the command is issued.

»This parameter is not supported for .NET clients if NMQ_MQ_LIB is set to `managed`, so that the client uses managed WebSphere MQ problem diagnostics.«

Return codes

AMQ7024	Non-valid arguments supplied to the command.
AMQ8304	Nine concurrent traces (the maximum) already running.

Examples

This command enables tracing of processing flow from common services and the local queue manager for a queue manager called QM1 in WebSphere MQ for UNIX systems. Trace data is generated at the default level of detail.

```
strmqtrc -m QM1 -t csflows -t lqmflows -t parms
```

This command disables tracing of SSL activity on a queue manager called QM1. Other trace data is generated at the parms level of detail.

```
strmqtrc -m QM1 -x ssl -t parms
```

This command enables high-detail tracing of the processing flow for all components:

```
strmqtrc -t all -t detail
```

This command enables tracing when FDC KN346050 or FDC KN346080 occur on any process that is using queue manager QM1:

```
strmqtrc -m QM1 -b FDC=KN346050,KN346080
```

This command enables tracing when FDC KN34650 occurs, and stops tracing when FDC KN346080 occurs. In both cases the FDC must occur on a process that is using queue manager QM1:

```
strmqtrc -m QM1 -b FDC=KN346050 -c FDC=KN346080
```

The next examples use the -p and -m flags to show the following:

- **How a combination of parameters on an individual invocation of the command are interpreted by Websphere MQ as having a logical AND between them.**
- **How multiple invocations of the command, without an intervening enqmqtrc command, are interpreted by Websphere MQ as having a logical OR between them:**

1. **This command enables tracing for all threads that result from any executing process called amqxxx.exe:**

```
strmqtrc -p amqxxx.exe
```

2.
   - If you invoke the following command after the command in step 1, without an intervening endmqtrc command, then tracing is limited to all threads that result from any executing process called amqxxx.exe *and* that are using queue manager QM2:

     ```
     strmqtrc -p amqxxx.exe -m QM2
     ```

   - If you invoke the following command after the command in step 1, without an intervening endmqtrc command, then tracing is limited to all processes and threads that result from executing amqxxx.exe *or* that are using queue manager QM2:

     ```
     strmqtrc -m QM2
     ```

Related commands

dspmqtrc	Display formatted trace output
endmqtrc	End trace

Parent topic: The control commands

This build: January 26, 2011 11:55:43

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16100_

5.3. Comparing command sets

The tables in this appendix compare the facilities available from the different administration command sets, and state whether you can perform each function from within the WebSphere® MQ Explorer.

Note: The following tables do not apply to WebSphere MQ for z/OS® or WebSphere MQ for i5/OS®. For information on how to use PCF commands on z/OS and on i5/OS, see WebSphere MQ Programmable Command Formats and Administration Interface. For information on how to use MQSC commands on z/OS and on i5/OS, see WebSphere MQ Script (MQSC) Command Reference.

- Queue manager commands
- Command server commands
- Authority commands
- Cluster commands
- Authentication information commands
- Channel commands
- Listener commands
- Namelist commands
- Process commands
- Queue commands
- Service commands
- Other commands

Parent topic: WebSphere MQ control commands

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22580_

5.3.1. Queue manager commands

Table 1. Queue manager commands

Description	PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Queue Manager	Change Queue Manager	ALTER QMGR	No equivalent	Yes
Create queue manager	No equivalent	No equivalent	**crtmqm**	Yes
Delete queue manager	No equivalent	No equivalent	**dltmqm**	Yes
Inquire Queue Manager	Inquire Queue Manager	DISPLAY QMGR	No equivalent	Yes
Inquire Queue Manager Status	Inquire Queue Manager Status	DISPLAY QMSTATUS	**dspmq**	Yes
Ping Queue Manager	Ping Queue Manager	PING QMGR	No equivalent	No
Refresh Queue Manager	No equivalent	REFRESH QMGR	No equivalent	No
Reset Queue Manager	Reset Queue Manager	RESET QMGR	No equivalent	No
Start queue manager	No equivalent	No equivalent	**strmqm**	Yes
Stop queue manager	No equivalent	No equivalent	**endmqm**	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:11

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22590_

5.3.2. Command server commands

Table 1. Commands for command server administration

Description	PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Display command server	Inquire Queue Manager Status	DISPLAY QMSTATUS	**dspmqcsv**	Yes
Start command server	Change Queue Manager	ALTER QMGR	**strmqcsv**	Yes
Stop command server	No equivalent	No equivalent	**endmqcsv**	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:11

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22600_

5.3.3. Authority commands

Table 1. Commands for authority administration

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Delete authority record	No equivalent	**setmqaut**	Yes

Inquire authority records	No equivalent	**dmpmqaut**	Yes
Inquire entity authority	No equivalent	**dspmqaut**	Yes
Refresh Security	REFRESH SECURITY	No equivalent	Yes
Set authority record	No equivalent	**setmqaut**	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22610_

5.3.4. Cluster commands

Table 1. Cluster commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Inquire Cluster Queue Manager	DISPLAY CLUSQMGR	No equivalent	Yes
Refresh Cluster	REFRESH CLUSTER	No equivalent	Yes
Reset Cluster	RESET CLUSTER	No equivalent	No
Resume Queue Manager Cluster	RESUME QMGR	No equivalent	Yes
Suspend Queue Manager Cluster	SUSPEND QMGR	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22620_

5.3.5. Authentication information commands

Table 1. Authentication information commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Authentication Information Object	ALTER AUTHINFO	No equivalent	Yes
Copy Authentication Information Object	DEFINE AUTHINFO(x) LIKE(y)	No equivalent	Yes
Create Authentication Information Object	DEFINE AUTHINFO	No equivalent	Yes
Delete Authentication Information Object	DELETE AUTHINFO	No equivalent	Yes
Inquire Authentication Information Object	DISPLAY AUTHINFO	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22630_

5.3.6. Channel commands

Table 1. Channel commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?

Change Channel	ALTER CHANNEL	No equivalent	Yes
Copy Channel	DEFINE CHANNEL(x) LIKE(y)	No equivalent	Yes
Create Channel	DEFINE CHANNEL	No equivalent	Yes
Delete Channel	DELETE CHANNEL	No equivalent	Yes
Inquire Channel	DISPLAY CHANNEL	No equivalent	Yes
Inquire Channel Names	DISPLAY CHANNEL	No equivalent	Yes
Inquire Channel Status	DISPLAY CHSTATUS	No equivalent	Yes
Ping Channel	PING CHANNEL	No equivalent	Yes
Reset Channel	RESET CHANNEL	No equivalent	Yes
Resolve Channel	RESOLVE CHANNEL	No equivalent	Yes
Start Channel	START CHANNEL	**runmqchl**	Yes
Start Channel Initiator	START CHINIT	**runmqchi**	No
Stop Channel	STOP CHANNEL	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22640_

5.3.7. Listener commands

Table 1. Listener commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Listener	ALTER LISTENER	No equivalent	Yes
Copy Listener	DEFINE LISTENER(x) LIKE(y)	No equivalent	Yes
Create Listener	DEFINE LISTENER	No equivalent	Yes
Delete Listener	DELETE LISTENER	No equivalent	Yes
Inquire Listener	DISPLAY LISTENER	No equivalent	Yes
Inquire Listener Status	DISPLAY LSSTATUS	No equivalent	Yes
Start Channel Listener	START LISTENER [1]	**runmqlsr**	Yes
Stop Listener	STOP LISTENER	**endmqlsr** [2]	Yes

Notes:

1. Used with listener objects only
2. Stops all active listeners

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22650_

5.3.8. Namelist commands

Table 1. Namelist commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Namelist	ALTER NAMELIST	No equivalent	Yes
Copy Namelist	DEFINE NAMELIST(x) LIKE(y)	No equivalent	Yes
Create Namelist	DEFINE NAMELIST	No equivalent	Yes
Delete Namelist	DELETE NAMELIST	No equivalent	Yes

Inquire Namelist	DISPLAY NAMELIST	No equivalent	Yes
Inquire Namelist Names	DISPLAY NAMELIST	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22660_

5.3.9. Process commands

Table 1. Process commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Process	ALTER PROCESS	No equivalent	Yes
Copy Process	DEFINE PROCESS(x) LIKE(y)	No equivalent	Yes
Create Process	DEFINE PROCESS	No equivalent	Yes
Delete Process	DELETE PROCESS	No equivalent	Yes
Inquire Process	DISPLAY PROCESS	No equivalent	Yes
Inquire Process Names	DISPLAY PROCESS	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22670_

5.3.10. Queue commands

Table 1. Queue commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Queue	ALTER QLOCAL ALTER QALIAS ALTER QMODEL ALTER QREMOTE	No equivalent	Yes
Clear Queue	CLEAR QLOCAL	No equivalent	Yes
Copy Queue	DEFINE QLOCAL(x) LIKE(y) DEFINE QALIAS(x) LIKE(y) DEFINE QMODEL(x) LIKE(y) DEFINE QREMOTE(x) LIKE(y)	No equivalent	Yes
Create Queue	DEFINE QLOCAL DEFINE QALIAS DEFINE QMODEL DEFINE QREMOTE	No equivalent	Yes
Delete Queue	DELETE QLOCAL DELETE QALIAS DELETE QMODEL DELETE QREMOTE	No equivalent	Yes
Inquire Queue	DISPLAY QUEUE	No equivalent	Yes
Inquire Queue Names	DISPLAY QUEUE	No equivalent	Yes
Inquire Queue Status	DISPLAY QSTATUS	No equivalent	Yes
Reset Queue Statistics	No equivalent	No equivalent	No

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22680_

5.3.11. Service commands

Table 1. Service commands

PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Change Service	ALTER SERVICE	No equivalent	Yes
Copy Service	DEFINE SERVICE(x) LIKE(y)	No equivalent	Yes
Create Service	DEFINE SERVICE	No equivalent	Yes
Delete Service	DELETE SERVICE	No equivalent	Yes
Inquire Service	DISPLAY SERVICE	No equivalent	Yes
Inquire Service Status	DISPLAY SVSTATUS	No equivalent	Yes
Start Service	START SERVICE	No equivalent	Yes
Stop Service	STOP SERVICE	No equivalent	Yes

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22690_

5.3.12. Other commands

Table 1. Other commands

Description	PCF command	MQSC command	Control command	WebSphere® MQ Explorer equivalent?
Create conversion exit	No equivalent	No equivalent	**crtmqcvx**	No
Display files used by objects	No equivalent	No equivalent	**dspmqfls**	No
Display formatted trace	No equivalent	No equivalent	**dspmqtrc** [1]	No
Display version information	No equivalent	No equivalent	**dspmqver**	No
Display transactions	No equivalent	No equivalent	**dspmqtrn**	No
Dump log	No equivalent	No equivalent	**dmpmqlog**	No
End trace	No equivalent	No equivalent	**endmqtrc**	Yes
Escape	Escape	No equivalent	No equivalent	No
Record media image	No equivalent	No equivalent	**rcdmqimg**	No
Recreate media object	No equivalent	No equivalent	**rcrmqobj**	No
Resolve transactions	No equivalent	No equivalent	**rsvmqtrn**	No
Run client trigger monitor	No equivalent	No equivalent	**runmqtmc**	No
Run dead-letter queue handler	No equivalent	No equivalent	**runmqdlq**	No
Run MQSC commands	No equivalent	No equivalent	**runmqsc**	No
Run trigger monitor	No equivalent	No equivalent	**runmqtrm**	No
Set service	No equivalent	No equivalent	**setmqscp** [2]	No

connection points				
Start WebSphere MQ trace	No equivalent	No equivalent	**strmqtrc**	Yes
WebSphere MQ Services control	No equivalent	No equivalent	**amqmdain** [2]	No

Notes:

1. Not supported on WebSphere MQ for Windows.
2. Supported by WebSphere MQ for Windows only.

Parent topic: »Comparing command sets«

This build: January 26, 2011 11:56:13

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22700_

5.4. Managing keys and certificates

To manage keys, certificates, and certificate requests, use one of the following:

The gsk7cmd command

The gsk7cmd command is available on UNIX systems only.

The gsk7cmd command provides functions that are similar to those of the iKeyman GUI, described in WebSphere MQ Security. The gsk7cmd command provides a shell script to run iKeycmd.

The runmqckm command

The runmqckm command is available on Windows systems only.

The runmqckm command provides functions that are similar to those of the iKeyman GUI, described in WebSphere MQ Security.

The gsk7capicmd command

The gsk7capicmd command is available on Windows and UNIX systems.

The gsk7capicmd command provides functions that are similar to those of the iKeyman GUI, described in WebSphere MQ Security. The gsk7capicmd command provides a shell script to run GSKCapiCmd. The command syntax for gsk7capicmd is the same as the syntax for runmqckm.

»If you need to manage SSL certificates in a way that is FIPS compliant, use the gsk7capicmd command instead of the gsk7cmd or runmqckm commands. This is because the gsk7capicmd command supports stronger encryption than the other commands.«

Before you run gsk7capicmd on Windows, set your PATH environment variable to include the GSkit binary and library directories. For example, at the command line, enter:

```
set PATH=%PATH%;C:\Program Files\IBM\gsk7\bin;C:\Program Files\IBM\gsk7\lib
```

Use the gsk7cmd, runmqckm, and gsk7capicmd commands to do the following:

- **Create the type of CMS key database files that WebSphere® MQ requires**
- **Create certificate requests**
- **Import personal certificates**
- **Import CA certificates**
- **Manage self-signed certificates**

Both the gsk7cmd and runmqckm commands execute an underlying WebSphere MQ component called iKeycmd. A default properties file, `ikeycmd.properties`, is provided as a sample file that you can modify.

The gsk7capicmd command executes an underlying WebSphere MQ component called GSKCapiCmd. There is no default properties file provided.

Preparing to use the gsk7cmd and gsk7capicmd commands

gsk7cmd, runmqckm, and gsk7capicmd commands

gsk7cmd, runmqckm, and gsk7capicmd options

gsk7capicmd error codes
A table of the numeric error codes issued by gsk7capicmd, and what they mean.

Parent topic: WebSphere MQ control commands

This build: January 26, 2011 11:55:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16110_

5.4.1. Preparing to use the gsk7cmd and gsk7capicmd commands

The gsk7cmd command is available on UNIX systems only. If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

The gsk7capicmd command is available on both UNIX and Windows systems.

To run the gsk7cmd and gsk7capicmd command line interfaces, set your PATH environment variable to include /usr/bin and /bin. For example, on UNIX:

```
export PATH=$PATH:/usr/bin:/bin
```

Additionally, for the gsk7cmd command only, set the JAVA_HOME environment variable to one of the following:

AIX®	`export JAVA_HOME=/usr/mqm/ssl/jre`
HP-UX	`export JAVA_HOME=/opt/mqm/ssl/jre`
Linux	`export JAVA_HOME=/opt/mqm/ssl/jre`
Solaris	`export JAVA_HOME=/opt/mqm/ssl`

Parent topic: Managing keys and certificates

This build: January 26, 2011 11:55:43

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16120_

5.4.2. gsk7cmd, runmqckm, and gsk7capicmd commands

Each command specifies at least one *object*. Commands for PKCS #11 device operations might specify additional objects. Commands for key database, certificate, and certificate request objects also specify an *action*.

This section describes commands according to the object of the command. The object can be one of the following:

–keydb
Actions apply to a key database

–cert
Actions apply to a certificate

–certreq
Actions apply to a certificate request

–help
Displays help

–version
Displays version information

The following sections describe the actions that you can take on key database, certificate, and certificate request objects:

- Commands for a CMS key database only
- Commands for CMS or PKCS #12 key databases
- Commands for cryptographic device operations

See gsk7cmd, runmqckm, and gsk7capicmd options for a description of the options on these commands.

Commands for a CMS key database only

Commands for CMS or PKCS #12 key databases

Commands for cryptographic device operations

Parent topic: Managing keys and certificates

This build: January 26, 2011 11:55:43

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16130_

5.4.2.1. Commands for a CMS key database only

–keydb –changepw

Change the password for a CMS key database:

Using the gsk7cmd and runmqckm commands:

```
-keydb -changepw -db filename -pw password -new_pw new_password
    -expire days -stash
```

Using the gsk7capicmd command:

```
-keydb -changepw -db filename -pw password -new_pw new_password
    -expire days -stash -fips -strong
```

–keydb –create

Create a CMS key database:

Using the gsk7cmd and runmqckm commands:

```
-keydb -create -db filename -pw password -type cms -expire days -stash
```

Using the gsk7capicmd command:

```
-keydb -create -db filename -pw password -type cms -expire days -stash
    -fips -strong
```

–keydb –stashpw

Stash the password of a CMS key database into a file:

Using the gsk7cmd and runmqckm commands:

```
-keydb -stashpw -db filename -pw password
```

Using the gsk7capicmd command:

```
-keydb -stashpw -db filename -pw password -fips
```

–cert –getdefault

Get the default personal certificate:

Using the gsk7cmd and runmqckm commands:

```
-cert -getdefault -db filename -pw password
```

Using the gsk7capicmd command:

```
-cert -getdefault -db filename -pw password -fips
```

–cert –modify

Modify a certificate.

Note: Currently, the only field that can be modified is the Certificate Trust field.

Using the gsk7cmd and runmqckm commands:

```
-cert -modify -db filename -pw password -label label
    -trust enable | disable
```

Using the gsk7capicmd command:

```
-cert -modify -db filename -pw password -label label
    -trust enable | disable -fips
```

–cert –setdefault

Set the default personal certificate:

Using the gsk7cmd and runmqckm commands:

```
-cert -setdefault -db filename -pw password -label label
```

Using the gsk7capicmd command:

```
-cert -setdefault -db filename -pw password -label label -fips
```

Parent topic: gsk7cmd, runmqckm, and gsk7capicmd commands

This build: January 26, 2011 11:55:43

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16140_

5.4.2.2. Commands for CMS or PKCS #12 key databases

–keydb –changepw

Change the password for a key database:

For the gsk7cmd and runmqckm commands:

```
-keydb -changepw -db filename -pw password -new_pw new_password -expire days
```

For the gsk7capicmd command:

```
-keydb -changepw -db filename -pw password -new_pw new_password -expire days
    -fips -strong
```

–keydb –convert

For the gsk7cmd and runmqckm commands, convert the key database from one format to another:

```
-keydb -convert -db filename -pw password
    -old_format cms | pkcs12 -new_format cms
```

For the gsk7capicmd command, convert an old version CMS key database to the new version CMS key database:

```
-keydb -convert -db filename -pw password
    -new_db filename -new_pw password -strong -fips
```

–keydb –create

Create a key database:

Using the gsk7cmd and runmqckm commands:

```
-keydb -create -db filename -pw password -type cms | pkcs12
```

Using the gsk7capicmd command:

```
-keydb -create -db filename -pw password -type cms -fips -strong
```

–keydb –delete

Delete a key database:

```
-keydb -delete -db filename -pw password
```

–keydb –list

List currently-supported types of key database:

Using the gsk7cmd and runmqckm commands:

```
-keydb -list
```

Using the gsk7capicmd command:

```
-keydb -list -fips
```

–cert –add

Add a certificate from a file into a key database:

```
-cert -add -db filename -pw password -label label -file filename
    -format ascii | binary
```

Using the gsk7capicmd command:

```
-cert -add -db filename -pw password -label label -file filename
    -format ascii | binary -fips
```

–cert –create

Create a self-signed certificate:

Using the gsk7cmd or runmqckmcommands:

```
-cert -create -db filename -pw password -label label -dn distinguished_name
    -size 1024 | 512 -x509version 3 | 1 | 2 -expire days
```

Using the gsk7capicmd command:

```
-cert -create -db filename -pw password -label label -dn distinguished_name
    -size 2048 | 1024 | 512 -x509version 3 | 1 | 2 -expire days
    -fips -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

–cert –delete

Delete a certificate:

Using the gsk7cmd or runmqckmcommands:

```
-cert -delete -db filename -pw password -label label
```

Using the gsk7capicmd command:

```
-cert -delete -db filename -pw password -label label -fips
```

–cert –details

List the detailed information for a specific certificate:

Using the gsk7cmd or runmqckm commands:

```
-cert -details -db filename -pw password -label label
```

Using the gsk7capicmd command:

```
-cert -details -db filename -pw password -label label -fips
```

–cert –export

Export a personal certificate and its associated private key from a key database into a PKCS #12 file, or to another key database:

Using the gsk7cmd or runmqckm commands:

```
-cert -export -db filename -pw password -label label -type cms | pkcs12
    -target filename -target_pw password -target_type cms | pkcs12
```

Using the gsk7capicmd command:

```
-cert -export -db filename -pw password -label label -type cms | pkcs12
    -target filename -target_pw password -target_type cms | pkcs12
    -encryption strong | weak -fips
```

–cert –extract

Extract a certificate from a key database:

Using the gsk7cmd or runmqckm commands:

```
-cert -extract -db filename -pw password -label label -target filename
    -format ascii | binary
```

Using the gsk7capicmd command:

```
-cert -extract -db filename -pw password -label label -target filename
    -format ascii | binary -fips
```

–cert –import

Import a personal certificate from a key database:

For the gsk7cmd and runmqckm commands:

```
-cert -import -file filename -pw password -type pkcs12 -target filename
       -target_pw password -target_type cms -label label
```

The `-label` option is required and specifies the label of the certificate that is to be imported from the source key database.

The `-new_label` option is optional and allows the imported certificate to be given a different label in the target key database from the label in the source database.

For the gsk7capicmd command:

```
-cert -import -file filename -pw password -type cms -target filename
       -target_pw password -target_type cms -label label -fips
```

The `-label` option is required and specifies the label of the certificate that is to be imported from the source key database.

The `-new_label` option is optional and allows the imported certificate to be given a different label in the target key database from the label in the source database.

–cert –list

List all certificates in a key database:

For the gsk7cmd and runmqckm commands:

```
-cert -list all | personal | CA
     -db filename -pw password
```

For the gsk7capicmd command:

```
-cert -list all | personal | CA
     -db filename -pw password -fips
```

–cert –receive

Receive a certificate from a file:

For the gsk7cmd and runmqckm commands:

```
-cert -receive -file filename -db filename -pw password
     -format ascii | binary -default_cert yes | no
```

For the gsk7capicmd command:

```
-cert -receive -file filename -db filename -pw password
     -format ascii | binary -default_cert yes | no -fips
```

–cert –sign

Sign a certificate:

For the gsk7cmd and runmqckm commands:

```
-cert -sign -db filename -file filename  -pw password
   -label label -target filename
   -format ascii | binary -expire days
```

For the gsk7capicmd command:

```
-cert -sign -db filename -file filename  -pw password
   -label label -target filename
   -format ascii | binary -expire days -fips
   -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

–certreq –create

Create a certificate request:

For the gsk7cmd and runmqckm commands:

```
-certreq -create -db filename -pw password
     -label label -dn distinguished_name
     -size 1024 | 512 -file filename
```

For the gsk7capicmd command:

```
-certreq -create -db filename -pw password
    -label label -dn distinguished_name
    -size 2048 | 1024 | 512 -file filename -fips
    -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

–certreq –delete

Delete a certificate request:

For the gsk7cmd and runmqckm commands:

```
-certreq -delete -db filename -pw password -label label
```

For the gsk7capicmd command:

```
-certreq -delete -db filename -pw password -label label -fips
```

–certreq –details

List the detailed information of a specific certificate request:

For the gsk7cmd and runmqckm commands:

```
-certreq -details -db filename -pw password -label label
```

For the gsk7capicmd command:

```
-certreq -details -db filename -pw password -label label -fips
```

List the detailed information about a certificate request and show the full certificate request:

Using the gsk7cmd and runmqckm commands:

```
-certreq -details -showOID -db filename
    -pw password -label label
```

Using the gsk7capicmd command:

```
-certreq -details -showOID -db filename
    -pw password -label label -fips
```

–certreq –extract

Extract a certificate request from a certificate request database into a file:

For the gsk7cmd and runmqckm commands:

```
-certreq -extract -db filename -pw password
     -label label -target filename
```

Using the gsk7capicmd command:

```
-certreq -extract -db filename -pw password
     -label label -target filename -fips
```

–certreq –list

List all certificate requests in the certificate request database:

For the gsk7cmd and runmqckm commands:

```
-certreq -list -db filename -pw password
```

Using the gsk7capicmd command:

```
-certreq -list -db filename -pw password -fips
```

–certreq –recreate

Recreate a certificate request:

For the gsk7cmd and runmqckm commands:

```
-certreq -recreate -db filename -pw password
    -label label -target filename
```

Using the gsk7capicmd command:

```
-certreq -recreate -db filename -pw password
    -label label -target filename -fips
```

Parent topic: gsk7cmd, runmqckm, and gsk7capicmd commands

This build: January 26, 2011 11:55:44

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16150_

5.4.2.3. Commands for cryptographic device operations

–keydb –changepw

Change the password for a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-keydb -changepw -crypto module_name -tokenlabel token_label
     -pw password -new_pw new_password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-keydb -changepw -db filename -crypto module_name -tokenlabel token_label
     -pw password -new_pw new_password -fips -strong
```

–keydb –list

List currently-supported types of key database:

Using the gsk7cmd and runmqckm commands:

```
-keydb -list
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-keydb -list -fips
```

–cert –add

Add a certificate from a file to a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -add -crypto module_name -tokenlabel token_label
     -pw password -label label -file filename -format ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -add -crypto module_name -tokenlabel token_label
     -pw password -label label -file filename -format ascii | binary
     -fips
```

–cert –create

Create a self-signed certificate on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -create -crypto module_name -tokenlabel token_label
     -pw password -label label -dn distinguished_name -size 1024 | 512
     -x509version 3 | 1 | 2 -default_cert no | yes -expire days
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and

must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -create -crypto module_name -tokenlabel token_label
     -pw password -label label -dn distinguished_name
     -size 2048 | 1024 | 512 -x509version 3 | 1 | 2
     -default_cert no | yes -expire days
     -fips -sigalg md5 | sha1 | sha224 | sha256 | sha384
```

–cert –delete

Delete a certificate on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -delete -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -delete -crypto module_name -tokenlabel token_label
     -pw password -label label -fips
```

–cert –details

List the detailed information for a specific certificate on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -details -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -details -crypto module_name -tokenlabel token_label
     -pw password -label label -fips
```

List the detailed information and show the full certificate for a specific certificate on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -details -showOID -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -details -showOID -crypto module_name -tokenlabel token_label
     -pw password -label label -fips
```

–cert –extract

Extract a certificate from a key database:

Using the gsk7cmd and runmqckm commands:

```
-cert -extract -crypto module_name -tokenlabel token_label
     -pw password -label label -target filename -format ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -extract -crypto module_name -tokenlabel token_label
     -pw password -label label -target filename -format ascii | binary
     -fips
```

–cert –import

Import a certificate to a cryptographic device with secondary key database support:

Using the gsk7cmd and runmqckm commands:

```
-cert -import -db filename -pw password -label label -type cms
     -crypto module_name -tokenlabel token_label -pw password
     -secondaryDB filename -secondaryDBpw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -import -db filename -pw password -label label -type cms
     -crypto module_name -tokenlabel token_label -pw password
     -secondaryDB filename -secondaryDBpw password -fips
```

Import a PKCS #12 certificate to a cryptographic device with secondary key database support:

Using the gsk7cmd and runmqckm commands:

```
-cert -import -file filename -pw password -type pkcs12
     -crypto module_name -tokenlabel token_label -pw password
     -secondaryDB filename -secondaryDBpw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -import -file filename -pw password -type pkcs12
     -crypto module_name -tokenlabel token_label -pw password
     -secondaryDB filename -secondaryDBpw password -fips
```

–cert –list

List all certificates on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-cert -list all | personal | CA
     -crypto module_name -tokenlabel token_label -pw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -list all | personal | CA
     -crypto module_name -tokenlabel token_label -pw password -fips
```

–cert –receive

Receive a certificate from a file to a cryptographic device with secondary key database support:

Using the gsk7cmd and runmqckm commands:

```
-cert -receive -file filename -crypto module_name -tokenlabel token_label
     -pw password  -default_cert yes | no
     -secondaryDB filename -secondaryDBpw password -format ascii | binary
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman

program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-cert -receive -file filename -crypto module_name -tokenlabel token_label
     -pw password  -default_cert yes | no
     -secondaryDB filename -secondaryDBpw password -format ascii | binary
     -fips
```

–certreq –create

Create a certificate request on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-certreq -create -crypto module_name -tokenlabel token_label
     -pw password -label label -dn distinguished_name
     -size 1024 | 512 -file filename
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -create -crypto module_name -tokenlabel token_label
     -pw password -label label -dn distinguished_name
     -size 2048 | 1024 | 512 -file filename -fips
     -sigalg md5 | sha1 | sha224 | sha256 | sha384 | sha512
```

–certreq –delete

Delete a certificate request from a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-certreq -delete -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -delete -crypto module_name -tokenlabel token_label
     -pw password -label label -fips
```

–certreq –details

List the detailed information of a specific certificate request on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-certreq -details -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -details -crypto module_name -tokenlabel token_label
     -pw password -label label -fips
```

List the detailed information about a certificate request and show the full certificate request on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-certreq -details -showOID -crypto module_name -tokenlabel token_label
     -pw password -label label
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and

must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -details -showOID -crypto module_name -tokenlabel token_label
    -pw password -label label -fips
```

–certreq –extract

Extract a certificate request from a certificate request database on a cryptographic device into a file:

Using the gsk7cmd and runmqckm commands:

```
-certreq -extract -crypto module_name -tokenlabel token_label
    -pw password -label label -target filename
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -extract -crypto module_name -tokenlabel token_label
    -pw password -label label -target filename -fips
```

–certreq –list

List all certificate requests in the certificate request database on a cryptographic device:

Using the gsk7cmd and runmqckm commands:

```
-certreq -list -crypto module_name -tokenlabel token_label
    -pw password
```

If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.

Using the gsk7capicmd command:

```
-certreq -list -crypto module_name -tokenlabel token_label
    -pw password -fips
```

Parent topic: gsk7cmd, runmqckm, and gsk7capicmd commands

This build: January 26, 2011 11:55:44

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16160_

5.4.3. gsk7cmd, runmqckm, and gsk7capicmd options

Table 1 lists the options that can be present on the command line. Note that the meaning of an option can depend on the object and action specified in the command.

*Table 1. Options that can be used with **gsk7cmd**, **runmqckm** , and **gsk7capicmd***

Option	Description
`-crypto`	Name of the module to manage a PKCS #11 cryptographic device. The value after `-crypto` is optional if you specify the module name in the properties file. Properties files are provided for **gsk7cmd** and **runmqckm** only. If you are using certificates or keys stored on PKCS #11 cryptographic hardware, note that iKeycmd and iKeyman are 32-bit programs. External modules required for PKCS #11 support will be loaded into a 32-bit process, therefore you must have a 32-bit PKCS #11 library installed for the administration of cryptographic hardware, and must specify this library to iKeycmd or iKeyman. The HP Itanium platform is the only exception, as the iKeyman program is 64–bit on the HP Itanium platform.
`-db`	Fully qualified path name of a key database.
`-default_cert`	Sets a certificate as the default certificate. The value can be `yes` or `no`. The default is `no`.
`-dn`	X.500 distinguished name. The value is a string enclosed in double quotes, for example

	"CN=John Smith,O=IBM,OU=Test,C=GB". Note that only the CN attribute is required. **Note:** You can use multiple OU attributes in distinguished names when you create self-signed certificates. Add additional OU key and value pairs to the specified distinguished name. For example: "CN=weblinux.Raleigh.ibm.com,O=ibm,OU=IBM HTTP Server,OU=GSKit\\, Gold Coast,L=RTP,ST=NC,C=US"
-encryption	Strength of encryption used in certificate export command. The value can be strong or weak. The default is strong.
-expire	Expiration time in days of either a certificate or a database password. The default is 365 days for a certificate password. There is no default time for a database password: use the -expire option to set a database password expiration time explicitly.
-file	File name of a certificate or certificate request.
-fips	specifies that the command is run in FIPS mode. This mode disables the use of the BSafe cryptographic library. Only the ICC component is used and this component must be successfully initialized in FIPS mode. When in FIPS mode, the ICC component uses algorithms that have been FIPS 140-2 validated. If the ICC component does not initialize in FIPS mode, the **gsk7capicmd** command fails.
-format	Format of a certificate. The value can be ascii for Base64_encoded ASCII or binary for Binary DER data. The default is ascii.
-label	Label attached to a certificate or certificate request.
-new_format	New format of key database (applicable to **gsk7cmd** and **runmqckm** only)..
-new_label	Used on a certificate import command, this option allows a certificate to be imported with a different label from the label it had in the source key database.
-new_pw	New database password.
-old_format	Old format of key database (applicable to **gsk7cmd** and **runmqckm** only).
-pw	Password for the key database or PKCS #12 file.
-secondaryDB	Name of a secondary key database for PKCS #11 device operations.
-secondaryDBpw	Password for the secondary key database for PKCS #11 device operations.
-showOID	Displays the full certificate or certificate request.
-sigalg	The hashing algorithm used during the creation of a certificate request, a self-signed certificate, or the signing of a certificate. This hashing algorithm is used to create the signature associated with the newly-created certificate or certificate request. The value can be md5, sha1, sha224, sha256, sha384, or sha512. The default is sha1.
-size	Key size. The value can be 512, 1024, or 2048. The default is 1024.
-stash	Stash the key database password to a file.
-strong	Check that the password entered satisfies the minimum requirements for the passwords strength. The minimum requirements for a password are as follows: • The password must be a minimum length of 14 characters. • The password must contain a minimum of one lower case character, one upper case character, and one digit or special character. Special characters include the asterisk (*), the dollar sign ($), the number sign (#) and the percent sign (%). A space is classified as a special character. • Each character can only occur a maximum of three times in a password. • A maximum of two consecutive characters in the password can be identical. • All characters described above are in the standard ASCII printable character set within the range from 0x20 to 0x7E inclusive.
-target	Destination file or database.
-target_pw	Password for the key database if -target specifies a key database.
-target_type	Type of database specified by -target operand. See -type option for permitted values.
-tokenLabel	Label of a PKCS #11 cryptographic device.
-trust	Trust status of a CA certificate. The value can be enable or disable. The default is enable.
-type	Type of database. The value can be: • cms for a CMS key database • pkcs12 for a PKCS #12 file.
-x509version	Version of X.509 certificate to create. The value can be 1, 2, or 3. The default is 3.

Parent topic: Managing keys and certificates

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16170_

5.4.4. gsk7capicmd error codes

A table of the numeric error codes issued by gsk7capicmd, and what they mean.

Error code	Error Message
0	Success
1	Unknown error occurred
2	An ASN.1 encoding/decoding error occurred.
3	An error occurred while initializing ASN.1 encoder/decoder.
4	An ASN.1 encoding/decoding error occurred because of an out-of-range index or non-existent optional field.
5	A database error occurred.
6	An error occurred while opening the database file, check for file existence and permission.
7	An error occurred while re-opening the database file.
8	Database creation failed.
9	The database already exists.
10	An error occurred while deleting the database file.
11	The database could not be opened.
12	An error occurred while reading the database file.
13	An error occurred while writing data to the database file.
14	A database validation error occurred.
15	An invalid database version was encountered.
16	An invalid database password was encountered.
17	An invalid database file type was encountered.
18	The specified database has been corrupted.
19	An invalid password was provided or the key database has been tampered with or corrupted.
20	A database key entry integrity error occurred.
21	A duplicate certificate already exists in the database.
22	A duplicate key already exists in the database (Record ID).
23	A certificate with the same label already existed in the key database.
24	A duplicate key already exists in the database (Signature).
25	A duplicate key already exists in the database (Unsigned Certificate).
26	A duplicate key already exists in the database (Issuer and Serial Number).
27	A duplicate key already exists in the database (Subject Public Key Info).
28	A duplicate key already exists in the database (Unsigned CRL).
29	The label has been used in the database.
30	A password encryption error occurred.
31	An LDAP related error occurred. (LDAP is not supported by this program)
32	A cryptographic error occurred.
33	An encryption/decryption error occurred.
34	An invalid cryptographic algorithm was found.
35	An error occurred while signing data.
36	An error occurred while verifying data.
37	An error occurred while computing digest of data.
38	An invalid cryptographic parameter was found.
39	An unsupported cryptographic algorithm was encountered.
40	The specified input size is greater than the supported modulus size.
41	An unsupported modulus size was found.
42	A database validation error occurred.
43	Key entry validation failed.
44	A duplicate extension field exists.

45	The version of the key is wrong.
46	A required extension field does not exist.
47	The validity period does not include today or does not fall within its issuer's validity period
48	The validity period does not include today or does not fall within its issuer's validity period.
49	An error occurred while validating private key usage extension.
50	The issuer of the key was not found.
51	A required certificate extension is missing.
52	An invalid basic constraint extension was found.
53	The key signature validation failed.
54	The root key of the key is not trusted.
55	The key has been revoked.
56	An error occurred while validating authority key identifier extension.
57	An error occurred while validating private key usage extension.
58	An error occurred while validating subject alternative name extension.
59	An error occurred while validating issuer alternative name extension.
60	An error occurred while validating key usage extension.
61	An unknown critical extension was found.
62	An error occurred while validating key pair entries.
63	An error occurred while validating CRL.
64	A mutex error occurred.
65	An invalid parameter was found.
66	A null parameter or memory allocation error was encountered.
67	Number or size is too large or too small.
68	The old password is invalid.
69	The new password is invalid.
70	The password has expired.
71	A thread related error occurred.
72	An error occurred while creating threads.
73	An error occurred while a thread was waiting to exit.
74	An I/O error occurred.
75	An error occurred while loading CMS.
76	A cryptography hardware related error occurred.
77	The library initialization routine was not successfully called.
78	The internal database handle table is corrupted.
79	A memory allocation error occurred.
80	An unrecognized option was found.
81	An error occurred while getting time information.
82	Mutex creation error occurred.
83	An error occurred while opening message catalog.
84	An error occurred while opening error message catalog
85	A null file name was found.
86	An error occurred while opening files, check for file existence and permissions.
87	An error occurred while opening files to read.
88	An error occurred while opening files to write.
89	There is no such file.
90	The file cannot be opened because of its permission setting.
91	An error occurred while writing data to files.
92	An error occurred while deleting files.
93	Invalid Base64-encoded data was found.
94	An invalid Base64 message type was found.
95	An error occurred while encoding data with Base64 encoding rule.
96	An error occurred while decoding Base64-encoded data.
97	An error occurred while getting a distinguished name tag.
98	The required common name field is empty.
99	The required country or region name field is empty.
100	An invalid database handle was found.
101	

	The key database does not exist.
102	The request key pair database does not exist.
103	The password file does not exist.
104	The new password is identical to the old one.
105	No key was found in the key database.
106	No request key was found.
107	No trusted CA was found.
108	No request key was found for the certificate.
109	There is no private key in the key database.
110	There is no default key in the key database.
111	There is no private key in the key record.
112	There is no certificate in the key record.
113	There is no CRL entry.
114	An invalid key database file name was found.
115	An unrecognized private key type was found.
116	An invalid distinguished name input was found.
117	No key entry was found that has the specified key label.
118	The key label list has been corrupted.
119	The input data is not valid PKCS12 data.
120	The password is invalid or the PKCS12 data has been corrupted or been created with later version of PKCS12
121	An unrecognized key export type was found.
122	An unsupported password-based encryption algorithm was found.
123	An error occurred while converting the key ring file to a CMS key database.
124	An error occurred while converting the CMS key database to a key ring file.
125	An error occurred while creating a certificate for the certificate request.
126	A complete issuer chain cannot be built.
127	Invalid WEBDB data was found.
128	There is no data to be written to the key ring file.
129	The number of days that you entered extends beyond the permitted validity period.
130	The password is too short; it must consist of at least {0} characters.
131	A password must contain at least one numeric digit.
132	All characters in the password are either alphabetic or numeric characters.
133	An unrecognized or unsupported signature algorithm was specified.
134	An invalid database type was encountered.
135	The specified secondary key database is in use by another PKCS#11 device.
136	No secondary key database was specified.
137	The label does not exist on the PKCS#11 device.
138	Password required to access the PKCS#11 device.
139	Password not required to access the PKCS#11 device.
140	Unable to load the cryptographic library.
141	PKCS#11 is not supported for this operation.
142	An operation on a PKCS#11 device has failed.
143	The LDAP user is not a valid user. (LDAP is not supported by this program)
144	The LDAP user is not a valid user. (LDAP is not supported by this program)
145	The LDAP query failed. (LDAP is not supported by this program)
146	An invalid certificate chain was found.
147	The root certificate is not trusted.
148	A revoked certificate was encountered.
149	A cryptographic object function failed.
150	There is no certificate revocation list data source available.
151	There is no cryptographic token available.
152	FIPS mode is not available.
153	There is a conflict with the FIPS mode settings.
154	The password entered does not meet the minimum required strength.
200	There was a failure during initialization of the program.

201	Tokenization of the arguments passed to the GSKCapiCmd Program failed.
202	The object identified in the command is not a recognized object.
203	The action passed is not a known -keydb action.
204	The action passed is not a known -cert action.
205	The action passed is not a known -certreq action.
206	There is a tag missing for the requested command.
207	The value passed with the –version tag is not a recognized value.
208	The value passed with the –size tag is not a recognized value.
209	The value passed in with the –dn tag is not in the correct format.
210	The value passed in with the –format tag is not a recognized value.
211	There was an error associated with opening the file.
212	PKCS12 is not supported at this stage.
213	The cryptographic token you are trying to change the password for is not password protected.
214	PKCS12 is not supported at this stage.
215	The password entered does not meet the minimum required strength.
216	FIPS mode is not available.
217	The number of days you have entered as the expiry date is out of the allowed range.
218	Password strength failed the minimum requirements.
219	No Default certificate was found in the requested key database.
220	An invalid trust status was encountered.
221	An unsupported signature algorithm was encountered. At this stage only MD5 and SHA1 are supported.
222	PCKS11 not supported for that particular operation.
223	The action passed is not a known –random action.
224	A length than less than zero is not allowed.
225	When using the –strong tag the minimum length password is 14 characters.
226	When using the –strong tag the maximum length password is 300 characters.
227	The MD5 algorithm is not supported when in FIPS mode.
228	The site tag is not supported for the –cert –list command. This attribute is added for backward compatibility and potential future enhancement.
229	The value associated with the -ca tag is not recognized. The value must be either 'true' or 'false'.
230	The value passed in with the –type tag is not valid.
231	The value passed in with the –expire tag is below the allowed range.
232	The encryption algorithm used or requested is not supported.
233	The target already exists.

Parent topic: Managing keys and certificates

This build: January 26, 2011 11:56:22

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70200_

6. Configuring »and managing« WebSphere MQ

»Configure WebSphere MQ to meet your organization needs, perform operational tasks, and solve operational problems.«

»Configuring WebSphere MQ«
Change the behavior of WebSphere® MQ or an individual queue manager to suit your installation's needs.

»File system support«
Queue manager data is stored in the file system. A queue manager makes use of file system locking to prevent multiple instances of a multi-instance queue manager being active at the same time.

»Setting up security on UNIX systems and Windows«
Security considerations specific to UNIX systems and Windows

»Transactional support«
Enable your applications to work reliably with databases.

»The WebSphere MQ dead-letter queue handler«
What is a dead-letter queue, how are messages put on it, and how do you manage it?

»Availability, recovery and restart«
Make your applications highly available by maintaining queue availability if a queue manager fails, and recover messages after server or storage failure.

»Problem determination«
Problem determination is set of procedures to follow to handle problems experienced with WebSphere MQ.

»WebSphere MQ and UNIX System V IPC resources«
A queue manager uses some IPC resources. Use ipcs -a to find out what resources are being used.

»WebSphere MQ and UNIX Process Priority«
Recommended ways to set process priority *nice* values.

Parent topic: System Administration Guide

This build: January 26, 2011 11:54:46

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12420_

6.1. Configuring WebSphere MQ

Change the behavior of WebSphere® MQ or an individual queue manager to suit your installation's needs.

You change WebSphere MQ configuration information by changing the values specified on a set of configuration attributes (or parameters) that govern WebSphere MQ.

How you change this configuration information, and where WebSphere MQ stores your changes, is platform-specific:

- WebSphere MQ for Windows uses the WebSphere MQ Explorer to make changes to attribute information within the Windows Registry. You can also use amqmdain to set some Registry values, as described in amqmdain.
- Users on all other platforms change attribute values by editing the WebSphere MQ configuration files. On WebSphere MQ for Linux (x86 platform) the WebSphere MQ configuration files can be edited using the WebSphere MQ Explorer.

»Changing configuration information on Windows systems«
WebSphere MQ configuration information is stored in the Windows Registry. You edit configuration information using the WebSphere MQ Explorer.

»Changing configuration information on UNIX systems«
Configuration attributes are held in configuration files, at the level of the node and of the queue manager.

»Attributes for changing WebSphere MQ configuration information«
Modify the configuration of WebSphere MQ

»Changing queue manager configuration information«
The attributes described here modify the configuration of an individual queue manager. They override any settings for WebSphere MQ. On WebSphere MQ for Windows systems and on WebSphere MQ for Linux (x86 platform) systems, modify configuration information using the WebSphere MQ Explorer. On other systems, modify the information by editing the qm.ini configuration file.

»System and default objects«
Lists the system and default objects created by the crtmqm command

Parent topic: »Configuring and managing WebSphere MQ«

This build: January 26, 2011 11:54:46

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa12430_

6.1.1. Changing configuration information on Windows systems

WebSphere® MQ configuration information is stored in the Windows Registry. You edit configuration information using the WebSphere MQ Explorer.

»All WebSphere MQ configuration information is stored in the Windows Registry, except for client configuration options, which are held in the client configuration file. There is a simple, or close, correlation between the contents of the Windows Registry and the WebSphere MQ configuration files.«

You edit configuration information using the WebSphere MQ Explorer (or by using the amqmdain command). Do not try to edit the Registry system file directly as this might adversely affect the smooth running of both your WebSphere MQ system and your Windows operating system.

»Viewing configuration information«
View a description of Windows Registry keys used by WebSphere MQ.

Parent topic: »Configuring WebSphere MQ«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12440_

6.1.1.1. Viewing configuration information

»View a description of Windows Registry keys used by WebSphere MQ.«

»View a description of the keys used by WebSphere MQ in the Windows Registry by searching the WebSphere® MQ Help System for "Settings in the Windows Registry".«

»Access the WebSphere MQ Help System from:

- An icon in the Windows Start menu
- Help in the WebSphere MQ Explorer

«

Parent topic: »Changing configuration information on Windows systems«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12450_

6.1.2. Changing configuration information on UNIX systems

Configuration attributes are held in configuration files, at the level of the node and of the queue manager.

On UNIX platforms, you can change WebSphere® MQ configuration attributes within:

- A WebSphere MQ configuration file (mqs.ini) to effect changes for WebSphere MQ on the node as a whole. There is one mqs.ini file for each node.
- A queue manager configuration file (qm.ini) to effect changes for specific queue managers. There is one qm.ini file for each queue manager on the node.

»Client configuration options are held separately, in the client configuration file.«

A configuration file (or *stanza* file) contains one or more stanzas, which are groups of lines in the .ini file that together have a common function or define part of a system, such as log functions, channel functions, and installable services.

Because the WebSphere MQ configuration file is used to locate the data associated with queue managers, a nonexistent or incorrect configuration file can cause some or all MQSC commands to fail. Also, applications cannot connect to a queue manager that is not defined in the WebSphere MQ configuration file.

Any changes you make to a configuration file usually do not take effect until the next time the queue manager is started.

On Linux (x86 platform) systems you can edit configuration information from the WebSphere MQ Explorer.

»Editing configuration files«
Edit configuration files using commands or a standard text editor.

»The WebSphere MQ configuration file, mqs.ini«

»Queue manager configuration files, qm.ini«

Parent topic: »Configuring WebSphere MQ«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12460_

6.1.2.1. Editing configuration files

»Edit configuration files using commands or a standard text editor.«

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files either:

- Automatically, using commands that change the configuration of queue managers on the node
- Manually, using a standard text editor

You can edit the default values in the WebSphere® MQ configuration files after installation.

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

When you create a new queue manager:

- Back up the WebSphere MQ configuration file
- Back up the new queue manager configuration file

Comments can be included in configuration files by adding a ";" or a "#" character before the comment text. If you want to use a ";" or a "#" character without it representing a comment, you can prefix the character with a "\" character and it will be used as part of the configuration data.

»When do you need to edit a configuration file?«
Edit a configuration file to recover from backup, move a queue manager, change the default queue manager or to assist IBM support.

»Configuration file priorities«
The value of an attribute is defined in multiple places. Attributes set in commands take precedence over attributes in configuration files.

Parent topic: »Changing configuration information on UNIX systems«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12470_

6.1.2.1.1. When do you need to edit a configuration file?

»Edit a configuration file to recover from backup, move a queue manager, change the default queue manager or to assist IBM support.«

You might need to edit a configuration file if, for example:

- You lose a configuration file. (Recover from backup if you can.)
- You need to move one or more queue managers to a new directory.
- You need to change your default queue manager; this could happen if you accidentally delete the existing queue manager.
- You are advised to do so by your IBM® Support Center.

Parent topic: »Editing configuration files«

This build: January 26, 2011 11:54:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12480_

6.1.2.1.2. Configuration file priorities

»The value of an attribute is defined in multiple places. Attributes set in commands take precedence over attributes in configuration files.«

The attribute values of a configuration file are set according to the following priorities:

- Parameters entered on the command line take precedence over values defined in the configuration files
- Values defined in the qm.ini files take precedence over values defined in the mqs.ini file

Parent topic: »Editing configuration files«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12490_

6.1.2.2. The WebSphere MQ configuration file, mqs.ini

The WebSphere® MQ configuration file, mqs.ini, contains information relevant to all the queue managers on the node. It is created automatically during installation.

The mqs.ini file for WebSphere MQ for UNIX systems is in the `/var/mqm` directory. It contains:

- The names of the queue managers
- The name of the default queue manager
- The location of the files associated with each of them

The supplied `LogDefaults` stanza for a new WebSphere MQ Version 7.0 installation does not contain any explicit values for the attributes. The lack of an attribute means that the default for this value is used upon creation of a new queue manager

The values shown for the `LogDefaults` stanza in Figure 1 are the default values. Note that a value of zero for the `LogBufferPages` attribute means 512.

If you require a non-default value, you must explicitly specify that value in the `LogDefaults` stanza.

Figure 1. Example of a WebSphere MQ configuration file for UNIX systems

```
#*******************************************************************#
#* Module Name: mqs.ini                                              *#
#* Type       : WebSphere MQ Machine-wide Configuration File         *#
#* Function   : Define WebSphere MQ resources for an entire machine  *#
#*******************************************************************#
#* Notes      :                                                      *#
#* 1) This is the installation time default configuration            *#
#*                                                                   *#
#*******************************************************************#
AllQueueManagers:
#*******************************************************************#
#* The path to the qmgrs directory, below which queue manager data   *#
#* is stored                                                         *#
#*******************************************************************#
DefaultPrefix=/var/mqm

LogDefaults:
    LogPrimaryFiles=3
    LogSecondaryFiles=2
    LogFilePages=4096
    LogType=CIRCULAR
    LogBufferPages=0
    LogDefaultPath=/var/mqm/log

QueueManager:
    Name=saturn.queue.manager
    Prefix=/var/mqm
```

```
    Directory=saturn!queue!manager

QueueManager:
    Name=pluto.queue.manager
    Prefix=/var/mqm
    Directory=pluto!queue!manager

DefaultQueueManager:
    Name=saturn.queue.manager

ApiExitTemplate:
    Name=OurPayrollQueueAuditor
    Sequence=2
    Function=EntryPoint
    Module=/usr/ABC/auditor
    Data=123

ApiExitCommon:
    Name=MQPoliceman
    Sequence=1
    Function=EntryPoint
    Module=/usr/MQPolice/tmqp
    Data=CheckEverything
```

Parent topic: »Changing configuration information on UNIX systems«

This build: January 26, 2011 11:54:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12500_

6.1.2.3. Queue manager configuration files, qm.ini

A queue manager configuration file, qm.ini, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager. The qm.ini file is automatically created when the queue manager with which it is associated is created.

A qm.ini file is held in the root of the directory tree occupied by the queue manager. For example, the path and the name for a configuration file for a queue manager called QMNAME is:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

The queue manager name can be up to 48 characters in length. However, this does not guarantee that the name is valid or unique. Therefore, a directory name is generated based on the queue manager name. This process is known as *name transformation*. For a description, see Understanding WebSphere MQ file names.

Figure 1 shows how groups of attributes might be arranged in a queue manager configuration file in WebSphere® MQ for UNIX systems.

Figure 1. Example queue manager configuration file for WebSphere MQ for UNIX systems»

```
#* Module Name: qm.ini                                              *#
#* Type        : WebSphere MQ queue manager configuration file      *#
#  Function    : Define the configuration of a single queue manager *#
#*                                                                    *#
#********************************************************************#
#* Notes       :                                                    *#
#* 1) This file defines the configuration of the queue manager      *#
#*                                                                    *#
#********************************************************************#

ExitPath:
    ExitsDefaultPath=/var/mqm/exits
    ExitsDefaultPath64=/var/mqm/exits64

Service:
    Name=AuthorizationService
    EntryPoints=13

ServiceComponent:
    Service=AuthorizationService
    Name=MQSeries.UNIX.auth.service
    Module=/opt/mqm/bin/amqzfu [1]
    ComponentDataSize=0
```

```
Log:
    LogPrimaryFiles=3
    LogSecondaryFiles=2
    LogFilePages=4096
    LogType=CIRCULAR
    LogBufferPages=0
    LogPath=/var/mqm/log/saturn!queue!manager/

XAResourceManager:
    Name=DB2 Resource Manager Bank
    SwitchFile=/usr/bin/db2swit
    XAOpenString=MQBankDB
    XACloseString=
    ThreadOfControl=THREAD

Channels: 2
    MaxChannels=20
    MaxActiveChannels=100
    MQIBindType=STANDARD

TCP:
    KeepAlive = Yes
    SvrSndBuffSize=32768
    SvrRcvBuffSize=32768
    Connect_Timeout=0

QMErrorLog:
    ErrorLogSize=262144
    ExcludeMessage=7234
    SuppressMessage=9001,9002,9202
    SuppressInterval=30

ApiExitLocal:
    Name=ClientApplicationAPIchecker
    Sequence=3
    Function=EntryPoint
    Module=/usr/Dev/ClientAppChecker
    Data=9.20.176.20
```

Notes for Figure 1:

1. `/usr/mqm/bin/amqzfu` on AIX®
2. For more information on the Channel stanza, see the WebSphere MQ Intercommunication manual.
3. The value of zero for `LogBufferPages` gives a value of 512.

Parent topic: Changing configuration information on UNIX systems

This build: January 26, 2011 11:54:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12510_

6.1.3. Attributes for changing WebSphere MQ configuration information

Modify the configuration of WebSphere® MQ

On WebSphere MQ for Windows systems and on WebSphere MQ for Linux (x86 platform) systems, modify configuration information using the WebSphere MQ Explorer. On other systems, modify the information by editing the mqs.ini configuration file.

The following are detailed:

- All queue managers
- Default queue manager
- Exit properties
- Log defaults for WebSphere MQ
- Advanced Configuration and Power Interface (ACPI)
- API exits

- Queue managers

»All queue managers«
Use the `General` and `Extended` WebSphere MQ properties page from the WebSphere MQ Explorer, or the `AllQueueManagers` stanza in the mqs.ini file to specify the following information about all queue managers.

»Default queue manager«
Use the `General` WebSphere MQ properties page from the WebSphere MQ Explorer, or the `DefaultQueueManager` stanza in the mqs.ini file to specify the default queue manager.

»Exit properties«
Use the `Exits` WebSphere MQ properties page from the WebSphere MQ Explorer, or the `ExitProperties` stanza in the mqs.ini file to specify configuration options used by queue manager exit programs.

»Log defaults for WebSphere MQ«
Use the `Default log settings` WebSphere MQ properties page from the WebSphere MQ Explorer, or the `LogDefaults` stanza in the mqs.ini file to specify information about log defaults for all queue managers.

»Advanced Configuration and Power Interface (ACPI)«
Use the `ACPI` WebSphere MQ properties page from the WebSphere MQ Explorer, to specify how WebSphere MQ is to behave when the system receives a suspend request.

»API exits«
Use the WebSphere MQ Explorer or the amqmdain command to change the Registry entries for API exits.

»Queue managers«
There is one `QueueManager` stanza for every queue manager. Use the stanza to specify the location of the queue manager directory.

Parent topic: »Configuring WebSphere MQ«

This build: January 26, 2011 11:54:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12520_

6.1.3.1. All queue managers

»Use the `General` and `Extended` WebSphere® MQ properties page from the WebSphere MQ Explorer, or the `AllQueueManagers` stanza in the mqs.ini file to specify the following information about all queue managers.«

DefaultPrefix=*directory_name*

This attribute specifies the path to the qmgrs directory, within which the queue manager data is kept.

If you change the default prefix for the queue manager, replicate the directory structure that was created at installation time.

In particular, you must create the qmgrs structure. Stop WebSphere MQ before changing the default prefix, and restart WebSphere MQ only after you have moved the structures to the new location and changed the default prefix.

Note: Do not delete the `/var/mqm/errors` directory on UNIX systems, or the `\errors` directory on Windows systems.

As an alternative to changing the default prefix, you can use the environment variable MQSPREFIX to override the `DefaultPrefix` for the crtmqm command.

Because of operating system restrictions, keep the supplied path sufficiently short so that the sum of the path length and any queue manager name is a maximum of 70 characters long.

ConvEBCDICNewline=NL_TO_LF|TABLE|ISO

EBCDIC code pages contain a newline (NL) character that is not supported by ASCII code pages (although some ISO variants of ASCII contain an equivalent).

Use the ConvEBCDICNewline attribute to specify how WebSphere MQ is to convert the EBCDIC NL character into ASCII format.

NL_TO_LF

Convert the EBCDIC NL character (X'15') to the ASCII line feed character, LF (X'0A'), for all EBCDIC to ASCII conversions.

NL_TO_LF is the default.

TABLE

Convert the EBCDIC NL character according to the conversion tables used on your platform for all EBCDIC to ASCII conversions.

The effect of this type of conversion might vary from platform to platform and from language to language; even on the same platform, the behavior might vary if you use different CCSIDs.

ISO

Convert:

- **ISO CCSIDs using the TABLE method**
- **All other CCSIDs using the NL_TO_CF method**

Possible ISO CCSIDs are shown in Table 1.

Table 1. List of possible ISO CCSIDs

CCSID	Code Set
819	ISO8859-1
912	ISO8859-2
915	ISO8859-5
1089	ISO8859-6
813	ISO8859-7
916	ISO8859-8
920	ISO8859-9
1051	roman8

If the ASCII CCSID is not an ISO subset, ConvEBCDICNewline defaults to NL_TO_LF.

For more information about data conversion, see the WebSphere MQ Application Programming Guide.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12530_

6.1.3.2. Default queue manager

»Use the `General` WebSphere® MQ properties page from the WebSphere MQ Explorer, or the `DefaultQueueManager` stanza in the mqs.ini file to specify the default queue manager.«

Name=*default_queue_manager*

The default queue manager processes any commands for which a queue manager name is not explicitly specified. The `DefaultQueueManager` attribute is automatically updated if you create a new default queue manager. If you inadvertently create a new default queue manager and then want to revert to the original, alter the `DefaultQueueManager` attribute manually.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12550_

6.1.3.3. Exit properties

» Use the `Exits` WebSphere® MQ properties page from the WebSphere MQ Explorer, or the `ExitProperties` stanza in the mqs.ini file to specify configuration options used by queue manager exit programs.«

CLWLMode=SAFE|FAST

The cluster workload exit, CLWL, allows you to specify which cluster queue in the cluster to open in response to

an MQI call (MQOPEN, MQPUT, and so on). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

SAFE

Run the CLWL exit in a separate process from the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- The CLWL server process (amqzlwa0) fails.
- The queue manager restarts the CLWL server process.
- The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process can affect performance.

FAST

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are no problems with your CLWL exit, and you are particularly concerned about performance.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12560_

6.1.3.4. Log defaults for WebSphere MQ

»Use the `Default log settings` WebSphere® MQ properties page from the WebSphere MQ Explorer, or the `LogDefaults` stanza in the mqs.ini file to specify information about log defaults for all queue managers.«

If the stanza does not exist then the MQ defaults will be used. The log attributes are used as default values when you create a queue manager, but can be overridden if you specify the log attributes on the crtmqm command. See crtmqm for details of this command.

Once a queue manager has been created, the log attributes for that queue manager are taken from the settings described in Queue manager logs.

The default prefix (specified in All queue managers) and log path specified for the particular queue manager (specified in Queue manager logs) allow the queue manager and its log to be on different physical drives. This is the recommended method, although by default they are on the same drive.

For information about calculating log sizes, see Calculating the size of the log.

Note: The limits given in the following parameter list are limits set by WebSphere MQ. Operating system limits might reduce the maximum possible log size.

LogPrimaryFiles=3|*2-254* (Windows)|*2-510* (UNIX systems)

The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles=2|*1-253* (Windows)|*1-509* (UNIX systems)

The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX systems.

The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages=*number*

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX systems the minimum number of log file pages is 64, and on Windows the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType=CIRCULAR|LINEAR

The type of log to be used. The default is CIRCULAR.

CIRCULAR

Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See Circular logging for a fuller explanation of circular logging.

LINEAR

For both restart recovery and media or forward recovery (creating lost or damaged data by replaying the contents of the log).

See Linear logging for a fuller explanation of linear logging.

If you want to change the default, you can either edit the `LogType` attribute, or specify linear logging using the crtmqm command. You cannot change the logging method after a queue manager has been created.

LogBufferPages=0|*0-4096*

The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In WebSphere MQ Version 7.0 this is 512 (2048 KB).

If you specify a number between 1 and 17, the queue manager defaults to 18 (72 KB). If you specify a number between 18 and 4096, the queue manager uses the number specified to set the memory allocated.

LogDefaultPath=*directory_name*

The directory in which the log files for a queue manager reside. The directory resides on a local device to which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- `<DefaultPrefix>\log` for WebSphere MQ for Windows where `<DefaultPrefix>` is the value specified on the `DefaultPrefix` attribute on the `All Queue Managers` WebSphere MQ properties page. This value is set at install time.
- `/var/mqm/log` for WebSphere MQ for UNIX systems

Alternatively, you can specify the name of a directory on the crtmqm command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the crtmqm command, the value of the `LogDefaultPath` attribute in the mqs.ini file is used.

The queue manager name is appended to the directory name to ensure that multiple queue managers use different log directories.

When the queue manager is created, a `LogPath` value is created in the log attributes in the configuration information, giving the complete directory name for the queue manager's log. This value is used to locate the log when the queue manager is started or deleted.

LogWriteIntegrity=SingleWrite|DoubleWrite|TripleWrite

The method the logger uses to reliably write log records.

SingleWrite

Some hardware guarantees that, if a write operation writes a page and fails for any reason, a subsequent read of the same page into a buffer results in each byte in the buffer being either:

- **The same as before the write, or**
- **The byte that should have been written in the write operation**

On this type of hardware (for example, ssa write cache enabled), it is safe for the logger to write log records in a single write as the hardware assures full write integrity. This method provides the highest level of performance.

DoubleWrite

The DoubleWrite method was the default method used in WebSphere MQ V5.2 and is available for back-compatibility purposes only.

TripleWrite

This is the default method. Where hardware that assures write integrity is not available, write log records using the TripleWrite method because it provides full write integrity.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12570_

6.1.3.5. Advanced Configuration and Power Interface (ACPI)

Use the `ACPI` WebSphere® MQ properties page from the WebSphere MQ Explorer, to specify how WebSphere MQ is to behave when the system receives a suspend request.

Windows supports the Advanced Configuration and Power Interface (ACPI) standard. This enables Windows users with ACPI enabled hardware to stop and restart channels when the system enters and resumes from suspend mode.

Note that the settings specified in the `ACPI` WebSphere MQ properties page are applied only when the Alert Monitor is running. The Alert Monitor icon is present on the taskbar if the Alert Monitor is running.

DoDialog=Y | N

Displays the dialog at the time of a suspend request.

DenySuspend=Y | N

Denies the suspend request. This is used if DoDialog=N, or if DoDialog=Y and a dialog cannot be displayed, for example, because your laptop lid is closed.

CheckChannelsRunning=Y | N

Checks whether any channels are running. The outcome can determine the outcome of the other settings.

The following table outlines the effect of each combination of these parameters:

DoDialog	DenySuspend	CheckChannels Running	Action
N	N	N	Accept the suspend request.
N	N	Y	Accept the suspend request.
N	Y	N	Deny the suspend request.
N	Y	Y	If any channels are running deny the suspend request; if not accept the request.
Y	N	N	Display the dialog (see note below; accept the suspend request). This is the default.
Y	N	Y	If no channels are running accept the suspend request; if they are display the dialog (see note below; accept the request).
Y	Y	N	Display the dialog (see note below; deny the suspend request).
Y	Y	Y	If no channels are running accept the suspend request; if they are display the dialog (see note below; deny the request).

Note: In cases where the action is to display the dialog, if the dialog cannot be displayed (for example because your laptop lid is closed), the DenySuspend option is used to determine whether the suspend request is accepted or denied.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12580_

6.1.3.6. API exits

»Use the WebSphere® MQ Explorer or the amqmdain command to change the Registry entries for API exits.«

Use the `Exits` WebSphere MQ properties page from the WebSphere MQ Explorer, or the `ApiExitTemplate` and `ApiExitCommon` stanza in the mqs.ini file to identify API exit routines for all queue managers. On Windows systems, you can also use the amqmdain command to change the Registry entries for API exits. (To identify API exit routines for individual queue managers, you use the `ApiExitLocal` stanza, as described in API exits.)

For a complete description of the attributes for these stanzas, see Configuring API exits.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12590_

6.1.3.7. Queue managers

There is one `QueueManager` stanza for every queue manager. Use the stanza to specify the location of the queue manager directory.

On UNIX systems, there is one `QueueManager` stanza for every queue manager. These attributes specify the queue manager name, and the name of the directory containing the files associated with that queue manager. The name of the directory is based on the queue manager name, but is transformed if the queue manager name is not a valid file name. (See Understanding WebSphere MQ file names for more information about name transformation.)

On Windows systems, this information is held in the Registry. You cannot use the WebSphere® MQ Explorer to change it directly.

Name=*queue_manager_name*

The name of the queue manager.

Prefix=*prefix*

Where the queue manager files are stored. By default, this is the same as the value specified on the DefaultPrefix attribute of the All Queue Managers information.

Directory=*name*

The name of the subdirectory under the `<prefix>\QMGRS` directory where the queue manager files are stored. This name is based on the queue manager name, but can be transformed if there is a duplicate name or if the queue manager name is not a valid file name.

Parent topic: »Attributes for changing WebSphere MQ configuration information«

This build: January 26, 2011 11:54:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12600_

6.1.4. Changing queue manager configuration information

The attributes described here modify the configuration of an individual queue manager. They override any settings for WebSphere® MQ. On WebSphere MQ for Windows systems and on WebSphere MQ for Linux (x86 platform) systems, modify configuration information using the WebSphere MQ Explorer. On other systems, modify the information by editing the qm.ini configuration file.

»When you are defining one of these stanzas, you do not need to start each item on a new line. You can use either a semicolon (;) or a hash character (#) to indicate a comment.«

»Installable services«

»Queue manager logs«

»Restricted mode«
This option applies to UNIX and Linux systems only. The `RestrictedMode` stanza is set by the -g option on the crtmqm command. Do not change this stanza after the queue manager has been created. If you do not use the -g option, the stanza is not created in the `qm.ini` file.

»XA resource managers«

»Attributes of channels«
These attributes determine the configuration of a channel.

»TCP, LU62, NETBIOS, and SPX«
Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

»Exit path«
Use the `Exits` queue manager properties page from the WebSphere MQ Explorer, or the `ExitPath` stanza in the qm.ini file to specify the path for user exit programs on the queue manager system.

»Queue manager error logs«

»Queue manager default bind type«

»TLS and SSL«
Use the SSL stanza of the queue manager configuration file to configure how SSL or TLS channels on your queue manager.

»Exit properties«
Use the ExitPropertiesLocal stanza in the qm.ini file to specify information about exit properties on a queue manager. Alternatively, you can set it using amqmdain. You cannot set this property using the WebSphere MQ Explorer.

»Subpool«
This stanza is created by WebSphere MQ. Do not change it.

Parent topic: »Configuring WebSphere MQ«

This build: January 26, 2011 11:54:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12610_

6.1.4.1. Installable services

There are significant implications to changing installable services and their components. For this reason, the installable services are read-only in the WebSphere® MQ Explorer. To change installable services in on Windows systems, use regedit or on UNIX systems use the `Service` stanza in the qm.ini file.

For each component within a service, you must also specify the name and path of the module containing the code for that component. On UNIX systems, use the `ServiceComponent` stanza for this.

Name=AuthorizationService|NameService

The name of the required service.

AuthorizationService

For WebSphere MQ, the Authorization Service component is known as the Object Authority Manager, or OAM.

The `AuthorizationService` stanza and its associated `ServiceComponent` stanza are added automatically when the queue manager is created. Add other `ServiceComponent` stanzas manually.

NameService

No name service is provided by default. If you require a name service, you must add the `NameService` stanza manually.

EntryPoints=*number-of-entries*

The number of entry points defined for the service. This includes the initialization and termination entry points.

SecurityPolicy=Default|NTSIDsRequired (WebSphere MQ for Windows only)

The SecurityPolicy attribute applies only if the service specified is the default authorization service, that is, the OAM. The SecurityPolicy attribute allows you to specify the security policy for each queue manager. The possible values are:

Default

Use the default security policy to take effect. If a Windows security identifier (NT SID) is not passed to the OAM for a particular user ID, an attempt is made to obtain the appropriate SID by searching the relevant security databases.

NTSIDsRequired

Pass an NT SID to the OAM when performing security checks.

See Windows security identifiers (SIDs) for more information.

SharedBindingsUserId=*user-type*

The SharedBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The SharedBindingsUserId attribute is used with relation to shared bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id. For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER – Authenticate user. The possible values are:

Default

The value of the *UserIdentifier* field is set as the real user Id.

Real

The value of the *UserIdentifier* field is set as the real user Id.

Effective

The value of the *UserIdentifier* field is set as the effective user Id.

FastpathBindingsUserId=*user-type*

The FastpathBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The FastpathBindingsUserId attribute is used with relation to fastpath bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id. For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER – Authenticate user. The possible values are:

Default

The value of the *UserIdentifier* field is set as the real user Id.

Real

The value of the *UserIdentifier* field is set as the real user Id.

Effective

The value of the *UserIdentifier* field is set as the effective user Id.

IsolatedBindingsUserId =*user-type*

The IsolatedBindingsUserId attribute applies only if the service specified is the default authorization service, that is, the OAM. The IsolatedBindingsUserId attribute is used with relation to isolated bindings only. This value allows you to specify whether the *UserIdentifier* field in the *IdentityContext* structure, from the MQZ_AUTHENTICATE_USER function, is the effective user Id or the real user Id. For information on the MQZ_AUTHENTICATE_USER function, see MQZ_AUTHENTICATE_USER – Authenticate user. The possible values are:

Default

The value of the *UserIdentifier* field is set as the effective user Id.

Real

The value of the *UserIdentifier* field is set as the real user Id.

Effective

The value of the *UserIdentifier* field is set as the effective user Id.

For more information about installable services and components, see Installable services and components.

For more information about security services in general, see Setting up security on UNIX systems and Windows.

»Service components«

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12620_

6.1.4.1.1. Service components

You need to specify service component information when you add a new installable service. On Windows systems use regedit, and on UNIX systems use the `ServiceComponent` stanza in the qm.ini file.

The authorization service stanza is present by default, and the associated component, the OAM, is active.

Service=*service_name*

The name of the required service. This must match the value specified on the `Name` attribute of the Service configuration information.

Name=*component_name*

The descriptive name of the service component. This must be unique and contain only characters that are valid for the names of WebSphere® MQ objects (for example, queue names). This name occurs in operator messages generated by the service. We recommend that this name begins with a company trademark or similar distinguishing string.

Module=*module_name*

The name of the module to contain the code for this component. This must be a full path name.

ComponentDataSize=*size*

The size, in bytes, of the component data area passed to the component on each call. Specify zero if no component data is required.

For more information about installable services and components, see Installable services and components.

Parent topic: »Installable services«

This build: January 26, 2011 11:54:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12630_

6.1.4.2. Queue manager logs

Use the `Log` queue manager properties page from the WebSphere® MQ Explorer, or the `Log` stanza in the qm.ini file, to specify information about logging on this queue manager.

By default, these settings are inherited from the settings specified for the default log settings for the queue manager (described in Log defaults for WebSphere MQ). Change these settings only if you want to configure this queue manager in a different way.

For information about calculating log sizes, see Calculating the size of the log.

Note: The limits given in the following parameter list are set by WebSphere MQ. Operating system limits might reduce the maximum possible log size.

LogPrimaryFiles=3|*2-254* (Windows)|*2-510* (UNIX systems)

The log files allocated when the queue manager is created.

The minimum number of primary log files you can have is 2 and the maximum is 254 on Windows, or 510 on UNIX systems. The default is 3.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems, and must not be less than 3.

The value is examined when the queue manager is created or started. You can change it after the queue manager has been created. However, a change in the value is not effective until the queue manager is restarted, and the effect might not be immediate.

LogSecondaryFiles=2|*1-253* (Windows)|*1-509* (UNIX systems)

The log files allocated when the primary files are exhausted.

The minimum number of secondary log files is 1 and the maximum is 253 on Windows, or 509 on UNIX systems. The default number is 2.

The total number of primary and secondary log files must not exceed 255 on Windows, or 511 on UNIX systems,

and must not be less than 3.

The value is examined when the queue manager is started. You can change this value, but changes do not become effective until the queue manager is restarted, and even then the effect might not be immediate.

LogFilePages=*number*

The log data is held in a series of files called log files. The log file size is specified in units of 4 KB pages.

The default number of log file pages is 4096, giving a log file size of 16 MB.

On UNIX systems the minimum number of log file pages is 64, and on Windows the minimum number of log file pages is 32; in both cases the maximum number is 65 535.

Note: The size of the log files specified during queue manager creation cannot be changed for a queue manager.

LogType=CIRCULAR|LINEAR

The type of logging to be used by the queue manager. You cannot change the type of logging to be used once the queue manager has been created. Refer to the description of the LogType attribute in Log defaults for WebSphere MQ for information about creating a queue manager with the type of logging you require.

CIRCULAR

Start restart recovery using the log to roll back transactions that were in progress when the system stopped.

See Circular logging for a fuller explanation of circular logging.

LINEAR

For both restart recovery and media or forward recovery (creating lost or damaged data by replaying the contents of the log).

See Linear logging for a fuller explanation of linear logging.

LogBufferPages=0|*0-4096*

The amount of memory allocated to buffer records for writing, specifying the size of the buffers in units of 4 KB pages.

The minimum number of buffer pages is 18 and the maximum is 4096. Larger buffers lead to higher throughput, especially for larger messages.

If you specify 0 (the default), the queue manager selects the size. In WebSphere MQ Version 7.0 this is 512 (2048 KB).

If you specify a number between 1 and 17, the queue manager defaults to 18 (72 KB). If you specify a number between 18 and 4096, the queue manager uses the number specified to set the memory allocated.

»The value is examined when the queue manager is started. The value can be increased or decreased within the limits stated above. However, a change in the value is not effective until the next time the queue manager is started.«

LogPath=*directory_name*

The directory in which the log files for a queue manager reside. This must exist on a local device to which the queue manager can write and, preferably, on a different drive from the message queues. Specifying a different drive gives added protection in case of system failure.

The default is:

- `C:\Program Files\IBM\WebSphere MQ\log` in WebSphere MQ for Windows.
- `/var/mqm/log` in WebSphere MQ for UNIX systems.

You can specify the name of a directory on the crtmqm command using the -ld flag. When a queue manager is created, a directory is also created under the queue manager directory, and this is used to hold the log files. The name of this directory is based on the queue manager name. This ensures that the log file path is unique, and also that it conforms to any limitations on directory name lengths.

If you do not specify -ld on the crtmqm command, the value of the `LogDefaultPath` attribute is used.

In WebSphere MQ for UNIX systems, user ID mqm and group mqm must have full authorities to the log files. If you change the locations of these files, you must give these authorities yourself. This is not required if the log files are in the default locations supplied with the product.

LogWriteIntegrity=SingleWrite|DoubleWrite|TripleWrite

The method the logger uses to reliably write log records.

SingleWrite

Some hardware guarantees that, if a write operation writes a page and fails for any reason, a subsequent read of the same page into a buffer results in each byte in the buffer being either:

- The same as before the write, or

- The byte that should have been written in the write operation

On this type of hardware (for example; SAN or SSA write cache enabled), it is safe for the logger to write log records in a single write as the hardware assures full write integrity. This method provides the highest level of performance.

DoubleWrite

The DoubleWrite method was the default method used in WebSphere MQ V5.2 and is available for back-compatibility purposes only.

TripleWrite

This is the default method. Where hardware that assures write integrity is not available, write log records using the TripleWrite method because it provides full write integrity.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12640_

6.1.4.3. Restricted mode

This option applies to UNIX and Linux systems only. The `RestrictedMode` stanza is set by the -g option on the crtmqm command. Do not change this stanza after the queue manager has been created. If you do not use the -g option, the stanza is not created in the `qm.ini` file.

Within the Queue Manager Data Directory there are a limited number of directories under which WebSphere® MQ applications create files while they are connected to the WebSphere MQ queue manager. The following directories are granted world write access in order to allow applications to create files in them:

```
/var/mqm/sockets/<QMGRNAME>/@ipcc/ssem/<hostname>
/var/mqm/sockets/<QMGRNAME>/@app/ssem/<hostname>
/var/mqm/sockets/<QMGRNAME>/zsocketapp/ssem/<hostname>
```

where *<QMGRNAME>* is the name of the queue manager, and *<hostname>* is the host name.

On some systems, it might be unacceptable to grant all users write access to these directories, particularly those users who have no need to access the queue manager. By using Restricted mode, the permissions on the directories used to store Queue Manager Queue Data and the permissions on the System V IPC shared memory used to communicate with the queue manager are modified so that they can only be accessed by members of the specified ApplicationGroup.

To use Restricted mode for a queue manager:

- The creator of the queue manager must be in the *mqm* group and in the ApplicationGroup.
- The *mqm* user ID must be in the Application Group.
- Users that need to administer the queue manager must be in the *mqm* group and in the ApplicationGroup.
- Users that need to run WebSphere MQ applications must be in the ApplicationGroup.

Any MQCONN or MQCONNX call issued by a user who is not in the ApplicationGroup fails with reason code MQRC_Q_MGR_NOT_AVAILABLE.

Restricted mode operates with the WebSphere MQ Authorization Service; therefore users must be granted the authority to connect to WebSphere MQ, and access the WebSphere MQ resources that they require via the WebSphere MQ Authorization Service. For more information on configuring the WebSphere MQ Authorization Service, see Setting up security on UNIX systems and Windows.

WebSphere MQ Restricted mode must only be used where the control provided by the WebSphere MQ Authorization Service does not provide sufficient isolation of queue managers resources.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12650_

6.1.4.4. XA resource managers

Use the `XA resource manager` queue manager properties page from the WebSphere® MQ Explorer, or the `XAResourceManager` stanza in the qm.ini file, to specify the following information about the resource managers

involved in global units of work coordinated by the queue manager.

Add XA resource manager configuration information manually for each instance of a resource manager participating in global units of work; no default values are supplied.

See Database coordination for more information about resource manager attributes.

Name=*name* (mandatory)

This attribute identifies the resource manager instance.

The `Name` value can be up to 31 characters in length. You can use the name of the resource manager as defined in its XA-switch structure. However, if you are using more than one instance of the same resource manager, you must construct a unique name for each instance. You can ensure uniqueness by including the name of the database in the `Name` string, for example.

WebSphere MQ uses the `Name` value in messages and in output from the dspmqtrn command.

Do not change the name of a resource manager instance, or delete its entry from the configuration information, once the associated queue manager has started and the resource manager name is in effect.

SwitchFile=*name* (mandatory)

The fully-qualified name of the load file containing the resource manager's XA switch structure.

If you are using a 64-bit queue manager with 32-bit applications, the `name` value should contain only the base name of the load file containing the resource manager's XA switch structure.

The 32-bit file will be loaded into the application from the path specified by ExitsDefaultPath.

The 64-bit file will be loaded into the queue manager from the path specified by ExitsDefaultPath64.

XAOpenString=*string* (optional)

The string of data to be passed to the resource manager's xa_open entry point. The contents of the string depend on the resource manager itself. For example, the string could identify the database that this instance of the resource manager is to access. For more information about defining this attribute, see:

- Adding resource manager configuration information for DB2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix

and consult your resource manager documentation for the appropriate string.

XACloseString=*string* (optional)

The string of data to be passed to the resource manager's xa_close entry point. The contents of the string depend on the resource manager itself. For more information about defining this attribute, see:

- Adding resource manager configuration information for DB2
- Adding resource manager configuration information for Oracle
- Adding resource manager configuration information for Sybase
- Adding resource manager configuration information for Informix

and consult your database documentation for the appropriate string.

ThreadOfControl=THREAD|PROCESS

This attribute is mandatory for WebSphere MQ for Windows. The queue manager uses this value for serialization when it needs to call the resource manager from one of its own multithreaded processes.

THREAD

The resource manager is fully *thread aware*. In a multithreaded WebSphere MQ process, XA function calls can be made to the external resource manager from multiple threads at the same time.

PROCESS

The resource manager is not *thread safe*. In a multithreaded WebSphere MQ process, only one XA function call at a time can be made to the resource manager.

The `ThreadOfControl` entry does not apply to XA function calls issued by the queue manager in a multithreaded application process. In general, an application that has concurrent units of work on different threads requires this mode of operation to be supported by each of the resource managers.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa12660_

6.1.4.5. Attributes of channels

These attributes determine the configuration of a channel.

Use the `Channels` queue manager properties page from the WebSphere® MQ Explorer, or the `CHANNELS` stanza in the qm.ini file, to specify information about channels.

MaxChannels=100|*number*

»For z/OS, the maximum number of channels allowed, in the range 1 to 9 999. The default is 200.«
»For all other platforms, the value must be between 1 and 65 535. The default is 100. «

MaxActiveChannels=*MaxChannels_value*

»The maximum number of channels allowed to be active at any time. The default is the value specified for the `MaxChannels` attribute. «

MaxInitiators=3|*number*

»The maximum number of initiators. The default and maximum value is 3. «

»MQIBindType=FASTPATH|STANDARD«

The binding for applications:

FASTPATH

Channels connect using MQCONNX FASTPATH; there is no agent process.

»STANDARD«

»Channels connect using STANDARD.«

PipeLineLength=1|*number*

The maximum number of concurrent threads a channel will use. The default is 1. Any value greater than 1 is treated as 2.

When you use pipelining, configure the queue managers at both ends of the channel to have a *PipeLineLength* greater than 1.

Note: Pipelining is only effective for TCP/IP channels.

AdoptNewMCA=NO|SVR|SDR|RCVR|CLUSRCVR|ALL|FASTPATH

If WebSphere MQ receives a request to start a channel, but finds that an instance of the channel is already running, in some cases the existing channel instance must be stopped before the new one can start. The `AdoptNewMCA` attribute allows you to control which types of channels can be ended in this way.

If you specify the `AdoptNewMCA` attribute for a given channel type, but the new channel fails to start because a matching channel instance is already running:

1. **The new channel tries to stop the previous one by requesting it to end.**
2. **If the previous channel server does not respond to this request by the time the AdoptNewMCATimeout wait interval expires, the thread or process for the previous channel server is ended.**
3. **If the previous channel server has not ended after step 2, and after the AdoptNewMCATimeout wait interval expires for a second time, WebSphere MQ ends the channel with a `CHANNEL IN USE` error.**

The AdoptNewMCA functionality applies to server, sender, receiver and cluster-receiver channels. In the case of a sender or server channel, only one instance of a channel with a particular name can be running in the receiving queue manager. In the case of a receiver or cluster-receiver channel, multiple instances of a channel with a particular name might be running in the receiving queue manager, but only one instance can run at any one time from a particular remote queue manager.

Note: AdoptNewMCA is not supported on requester or server-connection channels.

Specify one or more values, separated by commas or blanks, from the following list:

NO

The `AdoptNewMCA` feature is not required. This is the default.

SVR

Adopt server channels.

SDR

Adopt sender channels.

RCVR

Adopt receiver channels.

CLUSRCVR

Adopt cluster receiver channels.

ALL

Adopt all channel types except FASTPATH channels.

FASTPATH

Adopt the channel if it is a FASTPATH channel. This happens only if the appropriate channel type is also specified, for example: `AdoptNewMCA=RCVR,SVR,FASTPATH`.

Attention!: The AdoptNewMCA attribute might behave in an unpredictable fashion with FASTPATH channels. Exercise great caution when enabling the AdoptNewMCA attribute for FASTPATH channels.

AdoptNewMCATimeout=60|1 – 3600

The amount of time, in seconds, that the new channel instance waits for the old channel instance to end. Specify a value in the range 1 – 3600. The default value is 60.

AdoptNewMCACheck=QM|ADDRESS|NAME|ALL

The type of checking required when enabling the `AdoptNewMCA` attribute. If possible, perform full checking to protect your channels from being shut down, inadvertently or maliciously. At the very least, check that the channel names match.

Specify one or more of the following values, separated by commas or blanks:

QM

Check that the queue manager names match.

ADDRESS

Check the communications address. For example, the TCP/IP address.

NAME

Check that the channel names match.

ALL

Check for matching queue manager names, the communications address, and for matching channel names.

The default is `AdoptNewMCACheck=NAME,ADDRESS,QM`.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12670_
»

6.1.4.6. TCP, LU62, NETBIOS, and SPX

Use these queue manager properties pages, or stanzas in the qm.ini file, to specify network protocol configuration parameters. They override the default attributes for channels.

»TCP«

»Use the `TCP` queue manager properties page from the WebSphere® MQ Explorer, or the `TCP` stanza in the qm.ini file, to specify Transmission Control Protocol/Internet Protocol (TCP/IP) configuration parameters.

Port=1414|*port_number*

The default port number, in decimal notation, for TCP/IP sessions. The *well known* port number for WebSphere MQ is 1414.

Library1=*DLLName1* (WebSphere MQ for Windows only)

The name of the TCP/IP sockets DLL.

The default is WSOCK32.

KeepAlive=»NO«|YES

Switch the KeepAlive function on or off. KeepAlive=YES causes TCP/IP to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

ListenerBacklog=number

Override the default number of outstanding requests for the TCP/IP listener.

When receiving on TCP/IP, a maximum number of outstanding connection requests is set. This can be considered to be a *backlog* of requests waiting on the TCP/IP port for the listener to accept the request. The default listener backlog values are shown in Table 1.

Table 1. Default outstanding connection requests (TCP)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5
Linux	100
Solaris	100
HP-UX	20
AIX® V4.2 or later	100
AIX V4.1 or earlier	10

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the TCP backlog, so the effective TCP backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 1, the TCP/IP connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and retries the connection at a later time.

»SvrSndBuffSize=»32768«|number«

»The size in bytes of the TCP/IP send buffer used by the server end of a client-connection server-connection channel.«

»SvrRcvBuffSize=»32768«|number«

»The size in bytes of the TCP/IP receive buffer used by the server end of a client-connection server-connection channel.«

»Connect_Timeout=»0«|number«

»The number of seconds before an attempt to connect the socket times out. The default value of zero specifies that there is no connect timeout.«

«

LU62 (WebSphere MQ for Windows only)

Use the `LU6.2` queue manager properties page from the WebSphere MQ Explorer, or the `LU62` stanza in the qm.ini file, to specify SNA LU 6.2 protocol configuration parameters.

TPName

The TP name to start on the remote site.

Library1=*DLLName 1*

The name of the APPC DLL.

The default value is WCPIC32.

Library2=*DLLName2*

The same as Library1, used if the code is stored in two separate libraries.

The default value is WCPIC32.

NETBIOS (WebSphere MQ for Windows only)

Use the `Netbios` queue manager properties page from the WebSphere MQ Explorer, or the `NETBIOS` stanza in the qm.ini file, to specify NetBIOS protocol configuration parameters.

LocalName=*name*

The name by which this machine is known on the LAN.

AdapterNum=0|*adapter_number*

The number of the LAN adapter. The default is adapter 0.

NumSess=1|*number_of_sessions*

The number of sessions to allocate. The default is 1.

NumCmds=1|*number_of_commands*

The number of commands to allocate. The default is 1.

NumNames=1|*number_of_names*

The number of names to allocate. The default is 1.

Library1=*DLLName1*

The name of the NetBIOS DLL.

The default value is NETAPI32.

SPX (WebSphere MQ for Windows only)

Use the `SPX` queue manager properties page from the WebSphere MQ Explorer, or the `SPX` stanza in the qm.ini file, to specify SPX protocol configuration parameters.

Socket=5E86|*socket_number*

The SPX socket number in hexadecimal notation. The default is X'5E86'.

BoardNum=0|*adapter_number*

The LAN adapter number. The default is adapter 0.

KeepAlive=NO|YES

Switch the KeepAlive function on or off.

KeepAlive=YES causes SPX to check periodically that the other end of the connection is still available. If it is not, the channel is closed.

Library1=*DLLName1*

The name of the SPX DLL.

The default is WSOCK32.DLL.

Library2=*DLLName2*

The same as LibraryName1, used if the code is stored in two separate libraries.

The default is WSOCK32.DLL.

ListenerBacklog=number

Override the default number of outstanding requests for the SPX listener.

When receiving on SPX, a maximum number of outstanding connection requests is set. This can be considered to be a *backlog* of requests waiting on the SPX socket for the listener to accept the request. The default listener backlog values are shown in Table 2.

Table 2. Default outstanding connection requests (SPX)

Platform	Default ListenerBacklog value
Windows Server	100
Windows Workstation	5

Note: Some operating systems support a larger value than the default shown. Use this to avoid reaching the connection limit.

Conversely, some operating systems might limit the size of the SPX backlog, so the effective SPX backlog could be smaller than requested here.

If the backlog reaches the values shown in Table 2, the SPX connection is rejected and the channel cannot start. For message channels, this results in the channel going into a RETRY state and retrying the connection at a later time. For client connections, the client receives an MQRC_Q_MGR_NOT_AVAILABLE reason code from MQCONN and should retry the connection at a later time.

Parent topic: Changing queue manager configuration information

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12680_

6.1.4.7. Exit path

Use the `Exits` queue manager properties page from the WebSphere® MQ Explorer, or the `ExitPath` stanza in the qm.ini file to specify the path for user exit programs on the queue manager system.

ExitsDefaultPath=*string*

The ExitsDefaultPath attribute specifies the location of:

- **32-bit channel exits for clients**
- **32-bit channel exits and data conversion exits for servers**
- **Unqualified XA switch load files**

ExitsDefaultPath64=*string*

The ExitsDefaultPath64 attribute specifies the location of:

- 64-bit channel exits for clients
- 64-bit channel exits and data conversion exits for servers
- Unqualified XA switch load files

»API exits«

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12690_

6.1.4.7.1. API exits

Use the `Exits` queue manager properties page from the WebSphere® MQ Explorer, or the `ApiExitLocal` stanza in the qm.ini file to identify API exit routines for a queue manager. On Windows systems, you can also use the amqmdain command to change the Registry entries for API exits. (To identify API exit routines for all queue managers, you use the `ApiExitCommon` and `ApiExitTemplate` stanzas, as described in API exits.)

For a complete description of the attributes for these stanzas, see Configuring API exits.

Parent topic: »Exit path«

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12700_

6.1.4.8. Queue manager error logs

Use the `Extended` queue manager properties page from the WebSphere® MQ Explorer, or the `QMErrorLog` stanza in the qm.ini file to tailor the operation and contents of queue manager error logs.

ErrorLogSize=*maxsize*

Specifies the size of the queue manager error log at which it is copied to the backup. *maxsize* must be between 32768 and 2147483648 bytes. If ErrorLogSize is not specified, the default value of 262144 bytes (256KB) is used.

ExcludeMessage=*msgIds*

Specifies messages that are not to be written to the queue manager error log. *msgIds* contain a comma separated list of message id's from the following:

- »6254 - The system could not dynamically load shared library«
- 7163 - Job started message (i5/OS® only)
- 7234 - Number of messages loaded
- 9001 - Channel program ended normally
- 9002 - Channel program started
- 9202 - Remote host not available
- 9208 - Error on receive from host
- 9209 - Connection closed
- 9228 - Cannot start channel responder
- 9508 - Cannot connect to queue manager
- 9524 - Remote queue manager unavailable
- 9528 - User requested closure of channel
- 9558 - Remote Channel is not available
- 9999 - Channel program ended abnormally

SuppressMessage=*msgIds*

Specifies messages that will be written to the queue manager error log once only in a specified time interval. The time interval is specified by SuppressInterval. *msgIds* contain a comma separated list of message id's from the following:

7163 - Job started message (i5/OS only)

7234 - Number of messages loaded

9001 - Channel program ended normally

9002 - Channel program started

9202 - Remote host not available

9208 - Error on receive from host

9209 - Connection closed

9228 - Cannot start channel responder

9508 - Cannot connect to queue manager

9524 - Remote queue manager unavailable

9528 - User requested closure of channel

9558 - Remote Channel is not available

9999 - Channel program ended abnormally

If the same message id is specified in both SuppressMessage and ExcludeMessage, the message is excluded.

SuppressInterval=*length*

Specifies the time interval, in seconds, in which messages specified in SuppressMessage will be written to the queue manager error log once only. *length* must be between 1 and 86400 seconds. If SuppressInterval is not specified, the default value of 30 seconds is used.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12710_

6.1.4.9. Queue manager default bind type

Use the `Extended` queue manager properties page from the WebSphere® MQ Explorer, or the `Connection` stanza in the qm.ini file to specify the default bind type.

DefaultBindType=SHARED|ISOLATED

If DefaultBindType is set to ISOLATED, applications and the queue manager run in separate processes, and no resources are shared between them.

If DefaultBindType is set to SHARED, applications and the queue manager run in separate processes, however some resources are shared between them.

The default is SHARED.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12720_

6.1.4.10. TLS and SSL

»Use the SSL stanza of the queue manager configuration file to configure how SSL or TLS channels on your queue manager.«

Online Certificate Status Protocol (OCSP)

A certificate can contain an AuthorityInfoAccess extension. This extension specifies a server to be contacted through Online Certificate Status Protocol (OCSP). To allow SSL or TLS channels on your queue manager to use AuthorityInfoAccess extensions, ensure that the OCSP server named in them is available, is correctly configured, and is accessible over the network. For more information, see Working with revoked certificates.

CrlDistributionPoint (CDP)

A certificate can contain a CrlDistributionPoint extension. This extension contains a URL which identifies both the protocol used to download a certificate revocation list (CRL) and also the server to be contacted.

If you want to allow SSL or TLS channels on your queue manager to use CrlDistributionPoint extensions, ensure that the CDP server named in them is available, correctly configured, and accessible over the network.

The SSL Stanza

Use the SSL stanza in the qm.ini file to configure how SSL or TLS channels on your queue manager attempt to use the following facilities, and how they react if problems occur when using them.

In each of the following cases, if the value supplied is not one of the valid values listed, then

- The default value is taken.
- No error messages are written mentioning that an invalid value is specified.

AltGSKit

AltGSKit specifies whether to use the alternative GSKit Version 8 socket library for SSL connections. GSKit v8 allows secret key reset to interoperate with newer Java Runtime Environments but it is not certified for Common Criteria EAL4 compliance. Its valid values are:

- YES: SSL or TLS channels use the alternative GSKit library which is currently GSKit v 8.
- NO: SSL or TLS channels use the normal GSKit library. This is the default value.

CDPCheckExtensions

CDPCheckExtensions specifies whether SSL or TLS channels on this queue manager try to check CDP servers that are named in CrlDistributionPoint certificate extensions.

Valid values are:

- YES: SSL or TLS channels try to check CDP servers to determine whether a digital certificate is revoked. This value is the default.
- NO: SSL or TLS channels do not try to check CDP servers.

OCSPAuthentication

OCSPAuthentication specifies the action to be taken when a revocation status cannot be determined from an OCSP server.

If OCSP checking is enabled, an SSL or TLS channel program attempts to contact an OCSP server.

If the channel program is unable to contact any OCSP servers, or if no server knows the revocation status of the certificate, then the OCSPAuthentication parameter is consulted.

Valid values are:

- REQUIRED - Failure to determine the revocation status causes the connection to be closed with an error. This value is the default.
- WARN - Failure to determine the revocation status causes a warning message to be written in the queue manager error logs, but the connection is allowed to proceed.
- OPTIONAL - Failure to determine the revocation status allows the connection to proceed silently. No warnings or errors are given.

OCSPCheckExtensions

OCSPCheckExtensions specifies whether SSL or TLS channels on this queue manager try to check OCSP servers that are named in AuthorityInfoAccess certificate extensions.

Valid values are:

- YES: SSL or TLS channels try to check OCSP servers to determine whether a digital certificate is revoked. This value is the default.
- NO: SSL or TLS channels do not try to check OCSP servers.

SSLHTTPProxyName=*string*

The string is either the host name or network address of the HTTP Proxy server which is to be used by GSKit for OCSP checks. This address can be followed by an optional port number, enclosed in parentheses. If you do not specify the port number, the default HTTP port, 80, is used. On the HP-UX PA-RISC and Sun Solaris SPARC platforms, and for 32-bit clients on AIX, the network address can only be an IPv4 address; on other platforms it can be an IPv4 or IPv6 address.

This attribute might be necessary if a firewall prevents access to the URL of the OCSP responder.

StrictBasicConstraintsValidation=YES|NO

StrictBasicConstraintsValidation specifies whether intermediate CA certificates are required to include the isCA flag in the BasicConstraints extension. This provides the queue manager with protection against the possibility of impersonation by a party that uses a certificate that is not intended to be a CA certificate.

Valid values are:

- **YES: intermediate CA certificates are required to include the isCA flag.**
- **NO: intermediate CA certificates are not required to include the isCA flag. This value is the default.**

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:54:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12725_

6.1.4.11. Exit properties

Use the ExitPropertiesLocal stanza in the qm.ini file to specify information about exit properties on a queue manager. Alternatively, you can set it using amqmdain. You cannot set this property using the WebSphere® MQ Explorer.

By default, this setting is inherited from the setting specified for the exit properties for the queue manager (described in Exit properties). Change this setting only if you want to configure this queue manager in a different way.

CLWLMode=SAFE|FAST

The cluster workload exit, CLWL, allows you to specify which cluster queue in the cluster to open in response to an MQI call (MQOPEN, MQPUT, and so on). The CLWL exit runs either in FAST mode or SAFE mode depending on the value you specify on the CLWLMode attribute. If you omit the CLWLMode attribute, the cluster workload exit runs in SAFE mode.

SAFE

Run the CLWL exit in a separate process from the queue manager. This is the default.

If a problem arises with the user-written CLWL exit when running in SAFE mode, the following happens:

- **The CLWL server process (amqzlwa0) fails.**
- **The queue manager restarts the CLWL server process.**
- **The error is reported to you in the error log. If an MQI call is in progress, you receive notification in the form of a return code.**

The integrity of the queue manager is preserved.

Note: Running the CLWL exit in a separate process can affect performance.

FAST

Run the cluster exit inline in the queue manager process.

Specifying this option improves performance by avoiding the process switching costs associated with running in SAFE mode, but does so at the expense of queue manager integrity. You should only run the CLWL exit in FAST mode if you are convinced that there are no problems with your CLWL exit, and you are particularly concerned about performance.

If a problem arises when the CLWL exit is running in FAST mode, the queue manager will fail and you run the risk of the integrity of the queue manager being compromised.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70350_

6.1.4.12. Subpool

This stanza is created by WebSphere® MQ. Do not change it.

The stanza Subpool, and attribute ShortSubpoolName within that stanza, are written automatically by WebSphere MQ when you create a queue manager. WebSphere MQ chooses a value for ShortSubpoolName. Do not alter this value.

The name corresponds to a directory and symbolic link created inside the /var/mqm/sockets directory, which WebSphere MQ uses for internal communications between its running processes.

Parent topic: »Changing queue manager configuration information«

This build: January 26, 2011 11:56:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70360_

6.1.5. System and default objects

Lists the system and default objects created by the crtmqm command

When you create a queue manager using the crtmqm control command, the system objects and the default objects are created automatically.

- **The system objects are those WebSphere® MQ objects needed to operate a queue manager or channel.**
- **The default objects define all the attributes of an object. When you create an object, such as a local queue, any attributes that you do not specify explicitly are inherited from the default object.**

The following tables list the system and default objects created by crtmqm:

- **Table 1 lists the system and default queue objects.**
- **Table 2 lists the system and default topic objects.**
- **Table 3 lists the system and default channel objects.**
- **Table 4 lists the system and default authentication information objects.**
- **Table 5 lists the system and default listener objects.**
- **Table 6 lists the system and default namelist objects.**
- **Table 7 lists the system and default process objects.**
- **Table 8 lists the system and default service objects.**

Table 1. System and default objects: queues

Object name	Description
SYSTEM.ADMIN.ACCOUNTING.QUEUE	The queue that holds accounting monitoring data.
SYSTEM.ADMIN.ACTIVITY.QUEUE	The queue that holds returned activity reports.
SYSTEM.ADMIN.CHANNEL.EVENT	Event queue for channels.
»SYSTEM.ADMIN.COMMAND.EVENT«	»Event queue for command events.«
SYSTEM.ADMIN.COMMAND.QUEUE	Administration command queue. Used for remote MQSC commands and PCF commands.
SYSTEM.ADMIN.CONFIG.EVENT	Event queue for configuration events.
SYSTEM.ADMIN.PERFM.EVENT	Event queue for performance events.
SYSTEM.ADMIN.QMGR.EVENT	Event queue for queue manager events.
SYSTEM.ADMIN.STATISTICS.QUEUE	The queue that holds statistics monitoring data.
SYSTEM.ADMIN.TRACE.ROUTE.QUEUE	The queue that holds returned trace-route reply messages.
SYSTEM.AUTH.DATA.QUEUE	The queue that holds access control lists for the queue manager.
SYSTEM.CHANNEL.INITQ	Channel initiation queue.
SYSTEM.CHANNEL.SYNCQ	The queue that holds the synchronization data for channels.
SYSTEM.CICS.INITIATION.QUEUE	Default CICS® initiation queue.
SYSTEM.CLUSTER.COMMAND.QUEUE	The queue used to carry messages to the repository queue manager.
»SYSTEM.CLUSTER.HISTORY.QUEUE«	»The queue is used to store the history of cluster state information for service purposes.«
SYSTEM.CLUSTER.REPOSITORY.QUEUE	The queue used to store all repository information.
SYSTEM.CLUSTER.TRANSMIT.QUEUE	The transmission queue for all messages to all clusters.
SYSTEM.DEAD.LETTER.QUEUE	Dead-letter (undelivered-message) queue.
SYSTEM.DEFAULT.ALIAS.QUEUE	Default alias queue.

SYSTEM.DEFAULT.INITIATION.QUEUE	Default initiation queue.
SYSTEM.DEFAULT.LOCAL.QUEUE	Default local queue.
SYSTEM.DEFAULT.MODEL.QUEUE	Default model queue.
SYSTEM.DEFAULT.REMOTE.QUEUE	Default remote queue.
SYSTEM.MQEXPLORER.REPLY.MODEL	The WebSphere MQ Explorer reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to the WebSphere MQ Explorer.
SYSTEM.MQSC.REPLY.QUEUE	MQSC command reply-to queue. This is a model queue that creates a temporary dynamic queue for replies to remote MQSC commands.
SYSTEM.PENDING.DATA.QUEUE	Support deferred messages in JMS.

Table 2. System and default objects: topics

Object name	Description
SYSTEM.BASE.TOPIC	Base topic for ASPARENT resolution. If a particular topic has no parent administrative topic objects, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from this object.
SYSTEM.DEFAULT.TOPIC	Default topic definition.

Table 3. System and default objects: channels

Object name	Description
SYSTEM.AUTO.RECEIVER	Dynamic receiver channel.
SYSTEM.AUTO.SVRCONN	Dynamic server-connection channel.
SYSTEM.DEF.CLUSRCVR	Default receiver channel for the cluster, used to supply default values for any attributes not specified when a CLUSRCVR channel is created on a queue manager in the cluster.
SYSTEM.DEF.CLUSSDR	Default sender channel for the cluster, used to supply default values for any attributes not specified when a CLUSSDR channel is created on a queue manager in the cluster.
SYSTEM.DEF.RECEIVER	Default receiver channel.
SYSTEM.DEF.REQUESTER	Default requester channel.
SYSTEM.DEF.SENDER	Default sender channel.
SYSTEM.DEF.SERVER	Default server channel.
SYSTEM.DEF.SVRCONN	Default server-connection channel.
SYSTEM.DEF.CLNTCONN	Default client-connection channel.

Table 4. System and default objects: authentication information objects

Object name	Description
SYSTEM.DEFAULT.AUTHINFO.CRLLDAP	Default authentication information object for defining authentication information objects of type CRLLDAP.
»SYSTEM.DEFAULT.AUTHINFO.OCSP«	»Default authentication information object for defining authentication information objects of type OCSP.«

Table 5. System and default objects: listeners

Object name	Description
SYSTEM.DEFAULT.LISTENER.TCP	Default TCP listener.
SYSTEM.DEFAULT.LISTENER.LU62 (Windows only)	Default LU62 listener.
SYSTEM.DEFAULT.LISTENER.NETBIOS (Windows only)	Default NETBIOS listener.
SYSTEM.DEFAULT.LISTENER.SPX (Windows only)	Default SPX listener.

Table 6. System and default objects: namelists

Object name	Description
SYSTEM.DEFAULT.NAMELIST	Default namelist.

Table 7. System and default objects: processes

Object name	Description
SYSTEM.DEFAULT.PROCESS	Default process definition.

Table 8. System and default objects: services

Object name	Description
SYSTEM.DEFAULT.SERVICE	Default service.
SYSTEM.BROKER	Publish/subscribe broker

»Windows default configuration objects«

»SYSTEM.BASE.TOPIC«
Base topic for ASPARENT resolution. If a particular topic has no parent administrative topic objects, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from this object.

Parent topic: »Configuring WebSphere MQ«

This build: January 26, 2011 11:56:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22240_

6.1.5.1. Windows default configuration objects

On Windows systems, you set up a default configuration using the WebSphere® MQ Postcard application.

Note: You cannot set up a default configuration if other queue managers exist on your computer.

Many of the names used for the Windows default configuration objects involve the use of a short TCP/IP name. This is the TCP/IP name of the computer, without the domain part; for example the short TCP/IP name for the computer `mycomputer.hursley.ibm.com` is `mycomputer`. In all cases, where this name has to be truncated, if the last character is a period (.), it is removed.

Any characters within the short TCP/IP name that are not valid for WebSphere MQ object names (for example, hyphens) are replaced by an underscore character.

Valid characters for WebSphere MQ object names are: a to z, A to Z, 0 to 9, and the four special characters / % . and _.

The cluster name for the Windows default configuration is DEFAULT_CLUSTER.

If the queue manager is not a repository queue manager, the objects listed in Table 1 are created.

Table 1. Objects created by the Windows default configuration application

Object	Name
Queue manager	The short TCP/IP name prefixed with the characters QM_. The maximum length of the queue manager name is 48 characters. Names exceeding this limit are truncated at 48 characters. If the last character of the name is a period (.), this is replaced by a space ( ). The queue manager has a command server, a channel listener, and channel initiator associated with it. The channel listener listens on the standard WebSphere MQ port, port number 1414. Any other queue managers created on this machine must not use port 1414 while the default configuration queue manager still exists.
Generic cluster receiver channel	The short TCP/IP name prefixed with the characters TO_QM_. The maximum length of the generic cluster receiver name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).
Cluster sender channel	The cluster sender channel is initially created with the name TO_+QMNAME+. Once WebSphere MQ has established a connection to the repository queue manger for the default configuration cluster, this name is replaced with the name of the repository queue manager for the default configuration cluster, prefixed with the characters TO_. The maximum length of the cluster sender channel name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).
Local message queue	The local message queue is called `default`.
Local message queue for use by the WebSphere MQ Postcard application	The local message queue for use by the WebSphere MQ Postcard application is called `postcard`.
Server connection channel	The server connection channel allows clients to connect to the queue manager. Its name is the short TCP/IP name, prefixed with the characters S_.

	The maximum length of the server connection channel name is 20 characters. Names exceeding this limit are truncated at 20 characters. If the last character of the name is a period (.), this is replaced by a space ( ).

If the queue manager is a repository queue manager, the default configuration is similar to that described in Table 1, but with the following differences:

- **The queue manager is defined as a repository queue manager for the default configuration cluster.**
- **There is no cluster-sender channel defined.**
- **A local cluster queue that is the short TCP/IP name prefixed with the characters clq_default_ is created. The maximum length of this name is 48 characters. Names exceeding this length are truncated at 48 characters.**

If you request remote administration facilities, the server connection channel, SYSTEM.ADMIN.SVRCONN is also created.

Parent topic: »System and default objects«

This build: January 26, 2011 11:56:07

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22250_

6.1.5.2. SYSTEM.BASE.TOPIC

Base topic for ASPARENT resolution. If a particular topic has no parent administrative topic objects, or those parent objects also have ASPARENT, any remaining ASPARENT attributes are inherited from this object.

The default values of the SYSTEM.BASE.TOPIC are:

Table 1. Default values of SYSTEM.BASE.TOPIC

Parameter	Value
TOPICSTR	''
DEFPRTY	0
DEFPRESP	SYNC
DEFPSIST	NO
DESCR	'Base topic for resolving attributes'
DURSUB	YES
MDURMDL	SYSTEM.DURABLE.MODEL.QUEUE
MNDURMDL	SYSTEM.NDURABLE.MODEL.QUEUE
MASTER	YES
NPMSGDLV	ALLAVAIL
PMSGDLV	ALLDUR
PUB	ENABLE
SUB	ENABLE

If this object does not exist, its default values are still used by WebSphere® MQ for ASPARENT attributes that are not resolved by parent topics further up the topic tree.

Parent topic: »System and default objects«

This build: January 26, 2011 11:56:06

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22241

6.2. File system support

Queue manager data is stored in the file system. A queue manager makes use of file system locking to prevent multiple instances of a multi-instance queue manager being active at the same time.

Shared file systems

Shared file systems enable multiple systems to access the same physical storage device concurrently. Corruption would occur if multiple systems accessed the same physical storage device directly without some means of enforcing locking and concurrency control. Operating systems provide local file systems with locking and

concurrency control for local processes; network file systems provide locking and concurrency control for distributed systems. WebSphere MQ uses locking to prevent multiple instances of the same multi-instance queue manager being active at the same time.

Historically, networked file systems have not performed fast enough, or provided sufficient locking and concurrency control, to meet the requirements for logging messages. Today, networked file systems can provide good performance, and implementations of reliable network file system protocols such as *RFC 3530, Network File System (NFS) version 4 protocol*, meet the requirements for logging messages reliably.

Shared file systems and WebSphere MQ

Prior to release v7.0.1 WebSphere MQ does not support networked storage accessed as a shared file system. If queue manager data is placed on shared network storage, then you need to ensure the queue manager data is not accessed by another instance of the queue manager running at the same time.

From v7.0.1 onwards queue managers acquire a read/write lock to give them access to their data; only one instance of a queue manager can have its lock at a time.

Because not all the locking protocols of network file systems are robust, and because a file system might be configured for performance rather than data integrity, you must run the amqmfsck command to test whether a network file system will control access to queue manager data and logs correctly.

On Microsoft Windows and Unix systems, log files can also be placed in shared network storage, which is necessary for multi-instance queue managers. On IBM i, journals are used instead of log files, and journals cannot be shared. Multi-instance queue managers on IBM i use journal replication, or switchable journals, to make journals available between different queue manager instances.

»Requirements for shared file systems«
Shared files systems must provide data write integrity, guaranteed exclusive access to files and release locks on failure to work reliably with WebSphere® MQ.

»Sharing WebSphere MQ files«
Some WebSphere MQ files are accessed exclusively by an active queue manager, other files are shared.

»Directory structure on UNIX and Linux systems«
The WebSphere MQ directory structure on UNIX and Linux systems can be mapped to different file systems for easier management, better performance, and better reliability.

»Directory structure on Windows systems«
How to find queue manager configuration information and directories on Windows.

Parent topic: »Configuring and managing WebSphere MQ«

Related tasks
»Verifying shared file system locking«

This build: January 26, 2011 11:56:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70110_

6.2.1. Requirements for shared file systems

Shared files systems must provide data write integrity, guaranteed exclusive access to files and release locks on failure to work reliably with WebSphere® MQ.

There are three fundamental requirements that a shared file system must meet to log messages reliably:

1. Data write integrity.
   Data write integrity is sometimes called "Write through to disk on flush". The queue manager must be able to synchronize with data being successfully committed to the physical device. In a transactional system, you need to be sure that some writes have been safely committed before continuing with other processing.
2. Guaranteed exclusive access to files.
   In order to synchronize multiple queue managers, there needs to be a mechanism for a queue manager to obtain an exclusive lock on a file.
3. Release locks on failure.
   If a queue manager fails, or if there is a communication failure with the file system, files locked by the queue manager need to be unlocked and made available to other processes without waiting for the queue manager to be reconnected to the file system.
   Modern file systems, such as Network File System (NFS) Version 4, use leased locks to detect failures and then

release locks following a failure. Older file systems such as Network File System Version 3, which do not have a reliable mechanism to release locks after a failure, must not be used with multi-instance queue managers.

A shared file system *must* meet these requirements for WebSphere MQ to operate reliably. If it does not, the queue manager data and logs get corrupted when using the shared file system in a »multi-instance« queue manager configuration.

You must check whether the shared file system you plan to use meets these requirements. You must also check whether the file system is correctly configured for reliability. Shared file systems sometimes provide configuration options to improve performance at the expense of reliability.

Note: If you are using the NFS Version 4 file system as the shared file system, you must use hard mounts, synchronous writing and disable write caching, to fulfill these requirements.

»Under normal circumstances WebSphere MQ operates correctly with attribute caching and it is not necessary to disable caching, for example by setting NOAC on an NFS mount. Attribute caching can cause issues when multiple file system clients are contending for write access to the same file on the file system server, as the cached attributes used by each client might not be the same as those attributes on the server. An example of files accessed in this way are queue manager error logs for a multi-instance queue manager. The queue manager error logs might be written to by both an active and a standby queue manager instance and cached file attributes might cause the error logs to grow larger than expected, before rollover of the files occurs.«

To help to check the file system, run the task Verifying shared file system locking. This task checks if your shared file system meets requirements 2 and 3. You need to verify requirement 1 in your shared file system documentation, or by experimenting with logging data to the disk.

Disk faults can cause errors when writing to disk, which WebSphere MQ reports as First Failure Data Capture errors. You can run the file system checker for your operating system to check the shared file system for any disk faults. For example, on UNIX and Linux platforms the file system checker is called fsck. On Windows platforms the file system checker is called CHKDSK, or SCANDISK.

»Verifying shared file system locking«
Run amqmfsck to check obtaining and releasing file locks and concurrent writing on UNIX and IBM i systems.

Parent topic: »File system support«

This build: January 26, 2011 11:56:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70111_
»

6.2.1.1. Verifying shared file system locking

Run amqmfsck to check obtaining and releasing file locks and concurrent writing on UNIX and IBM i systems.

Before you begin

You need a machine with networked storage, and two other machines connected to it that have WebSphere MQ installed. You need to have administrator (root) authority to configure the file system, and be a WebSphere MQ Administrator to run amqmfsck.

About this task

Create the directory or directories on the networked storage that you are going to use to share queue manager data and logs. The directory owner needs to be a WebSphere MQ Administrator, or in other words, a member of the `mqm` group on UNIX. The user who runs the tests must have WebSphere MQ Administrator authority.

Use the example of exporting and mounting a file system in Create a multi-instance queue manager on Linux or Mirrored journal configuration on an ASP using ADDMQMJRN to help you through configuring the file system. Different file systems require different configuration steps. Refer to the file system documentation.

Procedure

1. Export the shared directory on the networked storage system and start the NFS daemon.
2. Mount the exported directory on the two WebSphere MQ servers.
3. Run amqmfsck, without any options, on each system to check basic locking
4. Run amqmfsck on both WebSphere MQ systems simultaneously, using the `-c` option, to test writing to the directory concurrently.
5. Run amqmfsck on both WebSphere MQ systems at the same time, using the `-w` option, to test waiting for and

releasing a lock on the directory concurrently.

Example

The topic, amqmfsck (file system check), includes examples of running amqmfsck.

Parent topic: »Requirements for shared file systems«

This build: January 26, 2011 11:56:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70120_

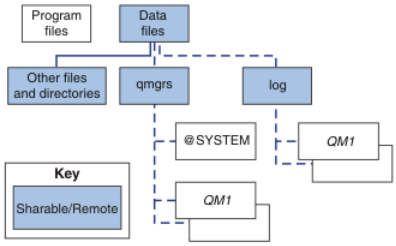
6.2.2. Sharing WebSphere MQ files

Some WebSphere® MQ files are accessed exclusively by an active queue manager, other files are shared.

WebSphere MQ files are split into program files and data files. Program files are typically installed locally on each server running WebSphere MQ. Queue managers share access to data files and directories in the default data directory. They require exclusive access to their own queue manager directory trees contained in each of the `qmgrs` and `log` directories shown in Figure 1.

Figure 1 is a high-level view of the WebSphere MQ directory structure showing the directories which can be shared between queue managers and made remote. The details vary by platform. The dotted lines indicate configurable paths.

Figure 1. Overall view of WebSphere MQ directory structure



Program files

The program files directory is typically left in the default location, is local, and shared by all the queue managers on the server.

Data files

The data files directory is typically local in the default location, `/var/mqm` on UNIX and configurable on installation on Windows. It is shared between queue managers. You can make the default location remote, but do not share it between different installations of WebSphere MQ. The `DefaultPrefix` attribute in the WebSphere MQ configuration points to this path.

`qmgrs`

From v7.0.1 there are two alternative ways to specify the location of queue manager data.

Using `Prefix`

The `Prefix` attribute specifies the location of the `qmgrs` directory. WebSphere MQ constructs the queue manager directory name from the queue manager name and creates it as a subdirectory of the `qmgrs` directory.

The `Prefix` attribute is located in the QueueManager stanza, and is inherited from the value in the `DefaultPrefix` attribute. By default, for administrative simplicity, queue managers typically share the same `qmgrs` directory.

The `QueueManager` stanza is in the registry in Windows, and in the `mqs.ini` file on UNIX platforms.

If you change the location of the `qmgrs` directory for any queue manager, you need to change the value of its `Prefix` attribute, unless the location of the queue manager data directory has been specified using the `DataPath` attribute.

The `Prefix` attribute for the `QM1` directory in Figure 1 for a UNIX platform is,

```
Prefix=/var/mqm
```

Using `DataPath`

The `DataPath` attribute specifies the location of the queue manager data directory.

Unlike the `Prefix` attribute that specifies an incomplete path to the queue manager data directory, the `DataPath` attribute specifies the complete path, including the name of the queue manager data directory.

The `DataPath` attribute, if it is specified, is located in the QueueManager stanza. If it has been specified, it takes precedence over any value in the `Prefix` attribute.

The `QueueManager` stanza is in the registry in Windows, and in the `mqs.ini` file on UNIX platforms.

If you change the location of the queue manager data directory for any queue manager, and you have specified a `DataPath` attribute, you need to change the value of the `DataPath` attribute.

The `DataPath` attribute for the `QM1` directory in Figure 1 for a UNIX platform is,

```
DataPath=/var/mqm/qmgrs/QM1
```

`log`

The log directory is specified separately for each queue manager in the `Log` stanza in the queue manager configuration. The queue manager configuration is in `qm.ini` on UNIX platforms. On Windows, queue managers that have a `DataPath` attribute also store the queue manager configuration in `qm.ini`, otherwise the queue manager configuration, which contains the `Log` stanza, is stored in the registry.

Other files and directories

Other files and directories, such as the directory containing trace files, and the common error log, are normally shared and kept on the local file system.

Up until v7.0.1, WebSphere MQ relied upon external management to guarantee queue managers exclusive access to the queue manager data and log files. From v7.0.1 onwards, with support of shared file systems, WebSphere MQ manages exclusive access to these files using file system locks, and allows only one instance of a particular queue manager to be active at a time.

When you start an instance of a particular queue manager it takes ownership of its queue manager directory. If you try to start another instance of the same queue manager on a different server, while the first queue manager is still running, the second instance will fail to start, and report the queue manager is running elsewhere. If the first queue manager had already stopped, then the second queue manager takes over ownership of the queue manager files and becomes the running queue manager.

You can automate the procedure of the second queue manager taking over from the first by starting a queue manager with the `strmqm -x` option that permits another queue manager to take over from it. The second queue manager then waits until the queue manager files are unlocked before attempting to take over ownership of the queue manager files, and start.

Parent topic: »File system support«

This build: January 26, 2011 11:56:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70112_

6.2.3. Directory structure on UNIX and Linux systems

The WebSphere MQ directory structure on UNIX and Linux systems can be mapped to different file systems for easier management, better performance, and better reliability.

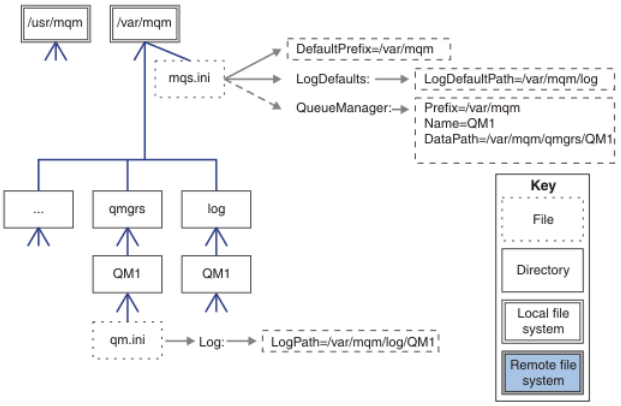
Use the flexible directory structure of WebSphere MQ to take advantage of shared file systems for running »multi-instance« queue managers.

Use the command crtmqm *QM1* to create the directory structure shown in Figure 1. It is a typical directory structure for a queue manager created on a WebSphere MQ UNIX system from v 7.0.1 onwards. Some directories, files and .ini attribute settings are omitted for clarity, and another queue manager name might be altered by mangling. The names of the file systems vary on different UNIX systems.

In a typical installation, every queue manager that you create points to common `log` and `qmgrs` directories on the local file system. In a »multi-instance« configuration, the `log` and `qmgrs` directories are on a network file system shared with another installation of WebSphere MQ.

Figure 1 shows the default configuration for WebSphere MQ v7.0.1 on AIX. For examples of alternative »multi-instance« configurations, see Example directory configurations on UNIX systems.

Figure 1. Example default WebSphere MQ v7.0.1 directory structure for UNIX systems



The product is installed into `/usr/mqm` on AIX and `/opt/mqm` on the other UNIX systems. The working directories are installed into the `/var/mqm` directory. These directory paths are fixed, although the paths vary between different UNIX systems.

The `log` and `qmgrs` directories are shown in their default locations as defined by the default values of the `LogDefaultPath` and `DefaultPrefix` attributes in the `mqs.ini` file. When a queue manager is created, by default the queue manager data directory is created in *`DefaultPrefix`*`/qmgrs`, and the log file directory in *`LogDefaultPath`*`/log`. `LogDefaultPath` and `DefaultPrefix` only effects where queue managers and log files are created by default. The actual location of a queue manager directory is saved in the `mqs.ini` file, and the location of the log file directory is saved in the `qm.ini` file.

The log file directory for a queue manager is defined in the `qm.ini` file in the `LogPath` attribute. Use the `-ld` option on the crtmqm command to set the `LogPath` attribute for a queue manager; for example, `crtmqm -ld` *`LogPath`* `QM1`. If you omit the `ld` parameter the value of `LogDefaultPath` is used instead.

The queue manager data directory is defined in the `DataPath` attribute in the `QueueManager` stanza in the `mqs.ini` file. Use the `-md` option on the crtmqm command to set the `DataPath` for a queue manager; for example, `crtmqm -md` *`DataPath`* `QM1`. If you omit the `md` parameter the value of the `DefaultPrefix` or `Prefix` attribute is used instead. `Prefix` takes precedence over `DefaultPrefix`.

Typically, create `QM1` specifying both the log and data directories in a single command.

```
crtmqm -md DataPath -ld LogPath QM1
```

You can modify the location of a queue manager log and data directories of an existing queue manager by editing the `DataPath` and `LogPath` attributes in the `qm.ini` file when the queue manager is stopped.

The path to the `errors` directory, like the paths to all the other directories in `/var/mqm`, is not modifiable. However the directories can be mounted on different file systems, or symbolically linked to different directories.

»Directory content on UNIX and Linuxsystems«
Content of the directories associated with a queue manager.

»Example directory configurations on UNIX systems«
Examples of alternative file system configurations on UNIX systems.

Parent topic: »File system support«

Related concepts
Example directory configurations on UNIX systems

Related reference
dspmqinf
addmqinf

This build: January 26, 2011 11:56:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70113_
»

6.2.3.1. Directory content on UNIX and Linuxsystems

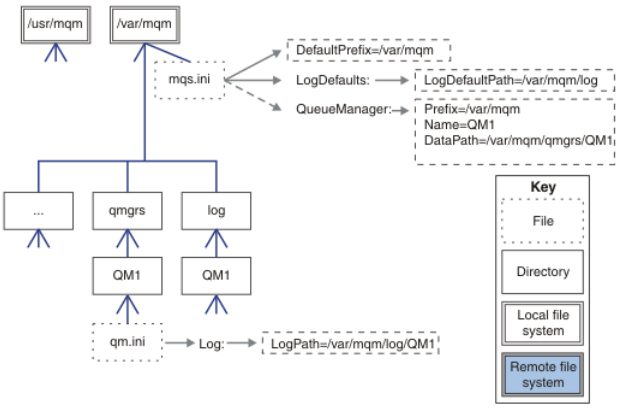
»Content of the directories associated with a queue manager.«

Figure 1 shows the general layout of the directories associated with a specific queue manager, QM1. The directories shown apply to the default installation. If you change this, the locations of the files and directories are modified accordingly. For information about the location of the product files, see one of the following:

- WebSphere MQ for AIX Quick Beginnings
- WebSphere MQ for HP-UX Quick Beginnings
- WebSphere MQ for Solaris Quick Beginnings
- WebSphere MQ for Linux Quick Beginnings

For information about alternative directory configurations, see File system support.

Figure 1. Example default WebSphere MQ v7.0.1 directory structure for UNIX systems



/var/mqm/

The /var/mqm directory contains configuration files and output directories that apply to a WebSphere MQ installation as a whole, and not to an individual queue manager.

Table 1. Documented content of the /var/mqm directory on UNIX systems

mqs.ini	WebSphere MQ installation wide configuration file, read when a queue manager starts. File path modifiable using the **AMQ_MQS_INI_LOCATION** environment variable. Ensure this is set and exported in the shell in which the **strmqm** command is run.
mqclient.ini	Default client configuration file read by WebSphere MQ client programs. File path modifiable using the **MQCLNTCF** environment variable.
service.env	Contains machine scope environment variables for a service process. File path fixed.
errors/	Machine scope error logs, and FFST files. Directory path fixed. See also, FFST: WebSphere MQ for UNIX systems.
sockets/	Contains information for each queue manager for system use only.
trace/	Trace files. Directory path fixed.
exits/ exits64/	Default directory containing user channel exit programs. Location modifiable in ApiExit stanzas in the mqs.ini file.

/var/mqm/qmgrs/*qmname*/

/var/mqm/qmgrs/*qmname*/ contains directories and files for a queue manager. The directory is locked for exclusive access by the active queue manager instance. The directory path is directly modifiable in the mqs.ini file, or by using the md option of the crtmqm command.

Table 2. Documented contents of the `/var/mqm/qmgrs/qmname` directory on UNIX systems

qm.ini	Queue manager configuration file, read when a queue manager starts.
errors/	Queue manager scope error logs. *qmname* = `@system` contains channel-related messages for an unknown or unavailable queue manager.
»@ipcc/AMQCLCHL.TAB«	Default client channel control table, created by the WebSphere MQ server, and read by WebSphere MQ client programs. File path modifiable using the **MQCHLLIB** and **MQCHLTAB** environment variables.
qmanager	Queue manager object file: `QMANAGER` Queue manager object catalog: `QMQMOBJCAT`
authinfo/ **channel/** **clntconn/** **listener/** **namelist/** **procdef/** **queues/** **services/** **topics/**	»Each object defined within the queue manager is associated with a file in these directories.« The file name approximately matches the definition name; see, Understanding WebSphere MQ file names.
...	Other directories used by WebSphere MQ, such as @ipcc, to be modified only by WebSphere MQ.

«

»

`/var/mqm/log/qmname/`

`/var/mqm/log/qmname/` contains the queue manager log files. The directory is locked for exclusive access by the active queue manager instance. The directory path is modifiable in the `qm.ini` file, or by using the ld option of the crtmqm command.

Table 3. Documented contents of the `/var/mqm/logsqmname` directory on UNIX systems

amqhlctl.lfh	Log control file.
active/	This directory contains the log files numbered S0000000.LOG, S0000001.LOG, S0000002.LOG, and so on.

«

Parent topic: »Directory structure on UNIX and Linux systems«

This build: January 26, 2011 11:56:08

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22270_

«

6.2.3.2. Example directory configurations on UNIX systems

Examples of alternative file system configurations on UNIX systems.

You can customize the WebSphere® MQ directory structure in various ways to achieve a number of different objectives.

- **Place the `qmgrs` and `log` directories on remote shared file systems to configure a »multi-instance« queue manager.**
- **Use separate file systems for the data and log directories, and allocate the directories to different disks, to improve performance by reducing I/O contention.**
- **Use faster storage devices for directories that have a greater impact on performance. Physical device latency is frequently a more important factor in the performance of persistent messaging than whether a device is mounted locally or remotely. The following list shows which directories are most and least performance**

sensitive.

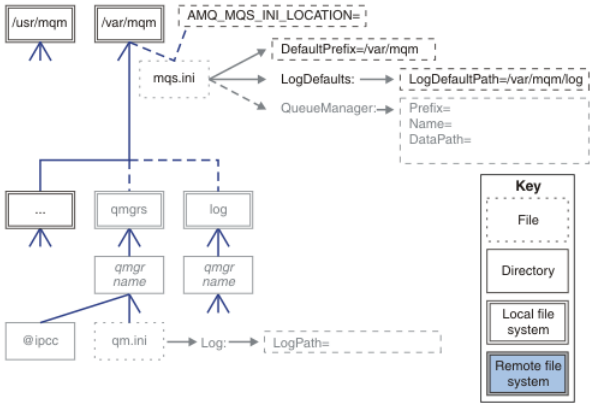
1. `log`
2. `qmgrs`
3. Other directories, including `/usr/mqm`

- Create the `qmgrs` and `log` directories on file systems that are allocated to storage with good resilience, such as a redundant disk array, for example.
- It is better to store the common error logs in `var/mqm/errors`, locally, rather than on a network file system, so that error relating to the network file system can be logged.

Figure 1 is a template from which alternative WebSphere MQ directory structures are derived. In the template, dotted lines represent paths that are configurable. In the examples, the dotted lines are replaced by solid lines that correspond to the configuration information stored in the AMQ_MQS_INI_LOCATION environment variable, and in the `mqs.ini` and `qm.ini` files.

Note: The path information is shown as it appears in the `mqs.ini` or `qm.ini` files. If you supply path parameters in the crtmqm command, omit the name of the queue manager directory: the queue manager name is added to the path by WebSphere MQ after it has been mangled.

Figure 1. Directory structure pattern template



Examples of configured directory structures are listed below. The first example shows a typical default directory structure for WebSphere MQ v7.0.1 created by issuing the crtmqm *QM1* command. The second example shows how a typical directory structure appears for a queue manager created using a WebSphere MQ release earlier than v7.0.1. The directory structure does not change.

Queue managers newly created in version 7.0.1 have a different configuration file to earlier releases of v7. If you need to remove the v7.0.1 fix pack to revert to v7.0.0.2, you need to recreate the configuration files. You might only need to use the Prefix attribute to define the path to the new queue manager data directory, or you might need to move the queue manager data directory and log directories to a different location. The safest way to reconfigure the queue manager is to save the queue manager data and log directories, delete and recreate the queue manager, and then replace the data and log directories in their new location, with the ones that have been saved.

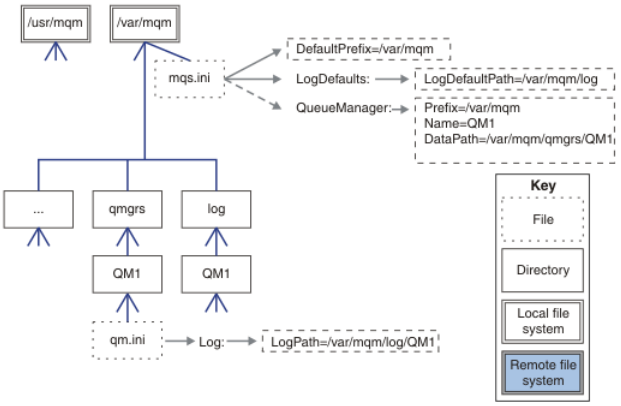
Typical directory structure for release v7.0.1 onwards

Figure 2 is the default directory structure created in v7.0.1 by issuing the command crtmqm *QM1*.

The `mqs.ini` file has a stanza for the QM1 queue manager, created by referring to the value of `DefaultPrefix`. The `Log` stanza in the `qm.ini` file has a value for `LogPath`, set by reference to `LogDefaultPath` in `mqs.ini`.

Use the optional crtmqm parameters to override the default values of `DataPath` and `LogPath`.

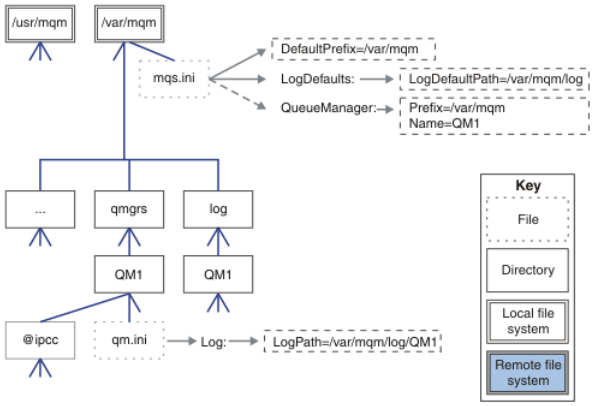
Figure 2. Example default WebSphere MQ v7.0.1 directory structure for UNIX systems



Typical directory structure for releases earlier than v7.0.1

The `DataPath` attribute did not exist before WebSphere MQ v7.0.1; the attribute is not present in the `mqs.ini` file. The location of the `qmgrs` directory was configured using the `Prefix` attribute. The location of individual directories could be configured by using symbolic links to point to different file system locations.

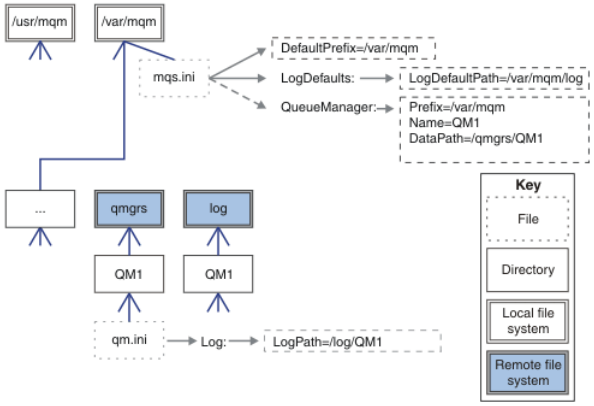
Figure 3. Typical directory structure for releases earlier than v7.0.1



Share default `qmgrs` and `log` directories (Release v7.0.1 onwards)

An alternative to Share everything (Release v7.0.1 onwards), is to share the `qmgrs` and `log` directories separately (Figure 4). In this configuration, there is no need to set `AMQ_MQS_INI_LOCATION` as the default mqs.ini is stored in the local `/var/mqm` file system. The files and directories, such as `mqclient.ini` and `mqserver.ini` are also not shared.

Figure 4. Share `qmgrs` and `log` directories

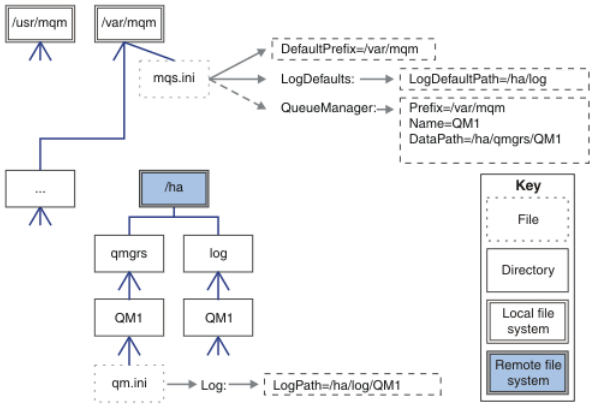


Share named `qmgrs` and `log` directories (Release v7.0.1 onwards)

The configuration in Figure 5 places the `log` and `qmgrs` in a common named remote shared file system called `/ha`. The same physical configuration can be created in two different ways.

1. **Set `LogDefaultPath=/ha` and then run the command, crtmqm `-md` *`/ha/qmgrs`* `QM1`. The result is exactly as illustrated in Figure 5.**
2. **Leave the default paths unchanged and then run the command, crtmqm `-ld` *`/ha/log`* `-md` *`/ha/qmgrs`* `QM1`.**

Figure 5. Share named `qmgrs` and `log` directories



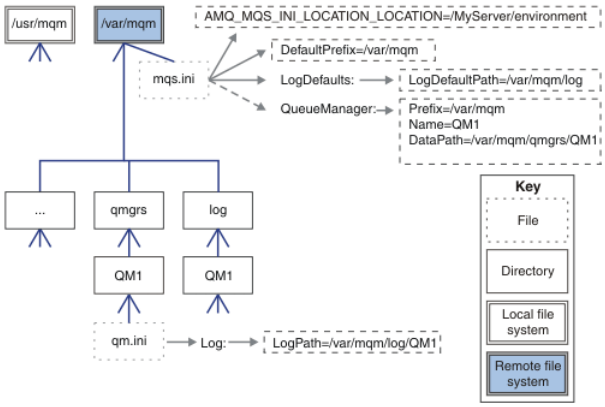
Share everything (Release v7.0.1 onwards)

Figure 6 is a simple configuration for system with fast networked file storage.

Mount `/var/mqm` as a remote shared file system. By default, when you start QM1, it looks for `/var/mqm`, finds it on the shared file system, and reads the `mqs.ini` file in `/var/mqm`. Rather than use the single `/var/mqm/mqs.ini` file for queue managers on all your servers, you can set the `AMQ_MQS_INI_LOCATION` environment variable on each server to point to different `mqs.ini` files.

Note: The contents the generic error file in `/var/mqm/errors/` shared between queue managers on different servers are interleaved.

Figure 6. Share everything



Parent topic: »Directory structure on UNIX and Linux systems«

This build: January 26, 2011 11:56:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70114_

6.2.4. Directory structure on Windows systems

How to find queue manager configuration information and directories on Windows.

The default directory for WebSphere MQ for Windows installation is:

32 bit

```
C:\Program Files\IBM\WebSphere MQ
```

64 bit

```
C:\Program Files (x86)\IBM\WebSphere MQ
```

The installation information is stored in the Windows registry. The registry key where WebSphere MQ information is stored is:

32 bit

```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSERIES\CurrentVersion\Configuration\
```

64 bit

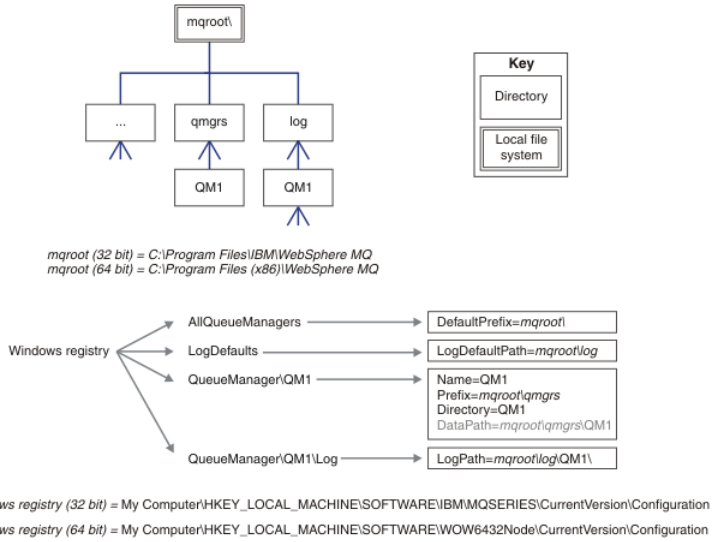
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\IBM\MQSERIES\CurrentVersion\Configuration\

The path that points to the WebSphere MQ data directory is stored in `AllQueueManagers` with the variable name *DefaultPrefix*, and the default directory for logs is stored in `LogDefaults` with the variable name *LogDefaultPath*. Queue manager data directories are created in *DefaultPrefix*\qmgrs*Qmgrname*. Queue manager logs are created in *LogDefaultPath**QmgrName*. See Figure 1.

If you define the queue manager data and log directories when installing WebSphere MQ, then *DefaultPrefix* and *LogDefaults* are updated with the customized path information.

DefaultPrefix and *LogDefaultPath* are only used to create a queue manager; a new key, QueueManager*QmgrName*, is created in the registry to store the queue manager configuration. Modify *DefaultPrefix* and *LogDefaultPath* at any time to change the queue manager data and log directories for new queue managers.

Figure 1. Default directory structure for Windows



Multi-instance queue managers

To configure a multi-instance queue manager, the log and data directories must be placed on networked storage, preferably on a different server to any of the servers that are running instances of the queue manager.

Two parameters are provided on the crtmqm command, -md and -ld, to make it easier specify the location of the queue manager data and log directories. The effect of specifying the -md parameter is fourfold:

1. The registry key QueueManager*QmgrName* contains a new variable, *DataPath*, which points to the queue manager data directory. Unlike the *Prefix* variable, the path includes the name of the queue manager directory.
2. The queue manager configuration information stored in the registry is reduced to *Name*, *Prefix*, *Directory* and *DataPath*.
3. All the other queue manager configuration information is stored in *DataPath*\qm.ini. Figure 2 and Figure 3 show the queue manager registry key and qm.ini configuration file for a queue manager with its data and log files located in networked storage. The queue manager installation is configured using customized program, data and log paths.
4. Queue manager status information is stored in *DataPath*\qmstatus.ini.

Figure 2. Example registry key

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager\QM2]
"Name"="QM2"
"Prefix"="C:\\IBM\\MQ\\MQ7\\Data"
"Directory"="QM2"
"DataPath"="N:\\MQHA\\Qmgrs\\QM2"
```

Figure 3. Example qm.ini file

```
#********************************************************************#
#* Module Name: qm.ini                                              *#
#* Type       : WebSphere MQ queue manager configuration file       *#
#  Function   : Define the configuration of a single queue manager *#
#*                                                                  *#
```

```
#*******************************************************************#
#* Notes       :                                                   *#
#* 1) This file defines the configuration of the queue manager     *#
#*                                                                 *#
#*******************************************************************#
ExitPath:
    ExitsDefaultPath=C:\IBM\MQ\MQ7\Data\exits
    ExitsDefaultPath64=C:\IBM\MQ\MQ7\Data\exits64
InstanceData:
    InstanceID=1242720143
    Startup=2
#*                                                                 *#
Log:
    LogPrimaryFiles=3
    LogSecondaryFiles=2
    LogFilePages=4096
    LogType=CIRCULAR
    LogBufferPages=0
    LogPath=N:\MQHA\log\QM2\
    LogWriteIntegrity=TripleWrite
Service:
    Name=AuthorizationService
    EntryPoints=13
ServiceComponent:
    Service=AuthorizationService
    Name=MQSeries.WindowsNT.auth.service
    Module=C:\IBM\MQ\MQ7\Programs\bin\amqzfu.dll
    ComponentDataSize=0
```

»Directory content«
Lists the location and content of WebSphere MQ directories.

Parent topic: »File system support«

This build: January 26, 2011 11:56:07

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22261_

6.2.4.1. Directory content

Lists the location and content of WebSphere MQ directories.

A WebSphere MQ configuration has three main sets of files and directories,

1. Executable, and other read-only files such as the Readme file, the WebSphere MQ Explorer plug-in and help files, and licenses files, that are only updated when maintenance is applied. These are described in Table 1.
2. Potentially modifiable files and directories that are not specific to a particular queue manager. These are described in Table 2.
3. Files and directories that are specific to each queue manager on a server. These are described in Table 1

Resource directories and files

The resource directories and files contain all the executable code and resources to run a queue manager. The variable, *FilePath*, in the WebSphere MQ Configuration registry key, contains the path to the resource directories.

Table 1. Directories and files in the FilePath directory

File path	Contents
FilePath\bin	Commands and DLLs
FilePath\bin64	Commands and DLLs (64 bit)
FilePath\doc	Wizard help files
FilePath\eclipse	Explorer and Explorer help Eclipse plug-ins
FilePath\gskit	Global security kit
FilePath\java	Java resources, including JRE
FilePath\licenses	License information

FilePath\Non_IBM_License	License information
FilePath\properties	Used internally
FilePath\Tivoli	
FilePath\tools	Development resources and samples
FilePath\Uninst	Used internally
FilePath\README.TXT	Readme

Directories not specific to a queue manager

Some directories contain files, such as trace files and error logs, that are not specific to a specific queue manager. The *DefaultPrefix* variable contains the path to these directories. *DefaultPrefix* is part of the AllQueueManagers registry key.

Table 2. Directories and files in DefaultPrefix directory

File path	Contents
DefaultPrefix\Config	Used internally
DefaultPrefix\conv	Data conversion tables
DefaultPrefix\errors	Non queue manager error logs, AMQERR*nn*.LOG
DefaultPrefix\exits	Channel exit programs
DefaultPrefix\exits64	Channel exit programs (64 bit)
DefaultPrefix\ipc	Not used
DefaultPrefix\Qmgrs	Described in Table 3
DefaultPrefix\trace	Trace files
DefaultPrefix\amqmjpse.txt	Used internally

Queue manager directories

When you create a queue manager, a new set of directories, specific to the queue manager, is created.

If you create a queue manager with the -md *filepath* parameter, the path is stored in the *DataPath* variable in the QueueManager/*QmgrName* registry key. If you create a queue manager without setting the -md *filepath* parameter, the queue manager directories are created in the path stored in *DefaultPrefix*, and the path is copied into the *Prefix* variable in the QueueManager/*QmgrName* registry key.

Table 3. Directories and files in DataPath and Prefix/Qmgrs/QmgrName directories

File path	Contents
DataPath\@ipcc	Default location for AMQCLCHL.TAB, the client connection table.
DataPath\authinfo	Used internally.
DataPath\channel	
DataPath\clntconn	
DataPath\errors	Error logs, AMQERR*nn*.LOG
DataPath\listener	Used internally.
DataPath\namelist	
DataPath\plugcomp	
DataPath\procdef	
DataPath\qmanager	
DataPath\queues	
DataPath\services	
DataPath\ssl	
DataPath\startprm	
DataPath\topic	
DataPath\active	
DataPath\active.dat	
DataPath\amqalchk.fil	
DataPath\master	
DataPath\master.dat	
DataPath\qm.ini [1]	Queue manager configuration
	Queue manager status

DataPath\qmstatus.ini[1]	
Prefix\Qmgrs*QmgrName*	Used internally
Prefix\Qmgrs\@SYSTEM	Not used
Prefix\Qmgrs\@SYSTEM\errors	

Parent topic: »Directory structure on Windows systems«

[1] Only exists for a queue manager with DataPath set in the queue manager stanza.

This build: January 26, 2011 11:56:07

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22262_

6.3. Setting up security on UNIX systems and Windows

»Security considerations specific to UNIX systems and Windows«

WebSphere® MQ queue managers transfer information that is potentially valuable, so you need to use an authority system to ensure that unauthorized users cannot access your queue managers. Consider the following types of security controls:

Who can administer WebSphere MQ

You can define the set of users who can issue commands to administer WebSphere MQ.

Who can use WebSphere MQ objects

You can define which users (usually applications) can use MQI calls and PCF commands to do the following:

- Who can connect to a queue manager.
- Who can access objects (queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects), and what type of access they have to those objects.
- Who can access WebSphere MQ messages.
- Who can access the context information associated with a message.

Channel security

You need to ensure that channels used to send messages to remote systems can access the required resources.

You can use standard operating facilities to grant access to program libraries, MQI link libraries, and commands. However, the directory containing queues and other queue manager data is private to WebSphere MQ; do not use standard operating system commands to grant or revoke authorizations to MQI resources.

»Common Criteria«
WebSphere MQ V7.0.1 has not been evaluated to Common Criteria EAL4.

»Authority to administer WebSphere MQ on UNIX and Windows systems«
WebSphere MQ administrators can use all WebSphere MQ commands and grant authorities for other users. When administrators issue commands to remote queue managers, they must have the required authority on the remote queue manager. Further considerations apply to Windows systems.

»Authority to work with WebSphere MQ objects on UNIX systems and Windows«
All objects are protected by WebSphere MQ, and principals must be given appropriate authority to access them. Different principals need different access rights to different objects.

»Connecting to WebSphere MQ using Terminal Services«
The "Create global objects" user right can cause problems if you are using Terminal Services.

»Configuring additional authority for Windows applications connecting to WebSphere MQ«
The account under which WebSphere MQ processes run might need additional authorization before SYNCHRONIZE access to application processes can be granted.

»Creating and managing groups«

»Using the OAM to control access to objects on UNIX systems and Windows«
The object authority manager (OAM) provides a command interface for granting and revoking authority to WebSphere MQ objects.

»Channel security«
User IDs for sending and receiving channels need access to various WebSphere MQ resources. For receiving channels, you can choose to use the user ID associated with the MCA or the message.

»How authorizations work«
The authorization specification tables in the topics in this section define precisely how the authorizations work and the restrictions that apply.

»Special considerations for security on Windows«
Some security functions behave differently on different version of Windows.

Parent topic: »Configuring and managing WebSphere MQ«

This build: January 26, 2011 11:54:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12730_

6.3.1. Common Criteria

»WebSphere MQ V7.0.1 has not been evaluated to Common Criteria EAL4.«

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:56:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22770_

6.3.2. Authority to administer WebSphere MQ on UNIX and Windows systems

WebSphere® MQ administrators can use all WebSphere MQ commands and grant authorities for other users. When administrators issue commands to remote queue managers, they must have the required authority on the remote queue manager. Further considerations apply to Windows systems.

WebSphere MQ administrators have authority to use all WebSphere MQ commands (including the commands to grant WebSphere MQ authorities for other users)

To be a WebSphere MQ administrator, you must be a member of a special group called the *mqm* group (or a member of the Administrators group on Windows systems). The mqm group is created automatically when WebSphere MQ is installed; add further users to the group to allow them to perform administration (including the root user on UNIX systems). All members of this group have access to all resources. This access can be revoked only by removing a user from the mqm group. Administrators can use control commands to administer WebSphere MQ. One of these control commands is setmqaut, which is used to grant authorities to other users to enable them to access WebSphere MQ resources.

Administrators can use the control command runmqsc to issue WebSphere MQ Script (MQSC) commands. When runmqsc is used in indirect mode to send MQSC commands to a remote queue manager, each MQSC command is encapsulated within an Escape PCF command. Administrators must have the required authorities for the MQSC commands to be processed by the remote queue manager. The WebSphere MQ Explorer issues PCF commands to perform administration tasks. Administrators require no additional authorities to use the WebSphere MQ Explorer to administer a queue manager on the local system. When the WebSphere MQ Explorer is used to administer a queue manager on another system, administrators must have the required authorities for the PCF commands to be processed by the remote queue manager.

For more information about authority checks when PCF and MQSC commands are processed, see

the following topics:

- For PCF commands that operate on queue managers, queues, processes, namelists, and authentication information objects, see Authority to work with WebSphere MQ objects. Refer to this section for the equivalent MQSC commands encapsulated within Escape PCF commands.
- For PCF commands that operate on channels, channel initiators, listeners, and clusters, see Channel security. Refer to this section for the equivalent MQSC commands encapsulated within Escape PCF commands.
- For MQSC commands that are processed by the command server on WebSphere MQ for z/OS®, see Command security and command resource security on z/OS.

»Additionally, on Windows systems, the SYSTEM account has full access to WebSphere MQ resources.«

On UNIX platforms, a special user ID of mqm is also created, for use by the product only. It must never be available to non-privileged users. All WebSphere MQ objects are owned by user ID mqm.

On Windows systems, members of the Administrators group can also administer any queue manager, »as can the SYSTEM account.« You can also create a domain mqm group on the domain controller that contains all privileged user IDs active within the domain, and add it to the local mqm group. Some commands, for example crtmqm, manipulate authorities on WebSphere MQ objects and so need authority to work with these objects (as described below). Members of the mqm group have authority to work with all objects, but there might be circumstances on Windows systems when authority is denied if you have a local user and a domain-authenticated user with the same name. This is described in Principals and groups.

Windows Vista and Windows Server 2008 introduces a User Account Control (UAC) feature, which restricts the actions users can perform on certain operating system facilities, even if they are members of the Administrators group. See User Account Control (UAC) on Windows Vista and Windows Server 2008 for more information. If your userid is in the Administrators group but not the mqm group you must use an elevated command prompt to issue MQ admin commands such as crtmqm, otherwise the error "AMQ7077: You are not authorized to perform the requested operation" is generated. To open an elevated command prompt, right-click the start menu item, or icon, for the command prompt, and select "Run as administrator".

You do not need to be a member of the mqm group to do the following:

- Issue commands from an application program that issues PCF commands, or MQSC commands within an Escape PCF command, unless the commands manipulate channel initiators. (These commands are described in Protecting channel initiator definitions).
- Issue MQI calls from an application program (unless you want to use the fastpath bindings on the MQCONNX call).
- Use the crtmqcvx command to create a fragment of code that performs data conversion on data type structures.
- Use the dspmq command to display queue managers.
- Use the dspmqtrc command to display WebSphere MQ formatted trace output.

A 12 character limitation applies to both group and user IDs.

UNIX platforms generally restrict the length of a user ID to 12 characters. AIX® Version 5.3 has raised this limit but WebSphere MQ continues to observe a 12 character restriction on all UNIX platforms. If you use a user ID of greater than 12 characters, WebSphere MQ replaces it with the value UNKNOWN. Do not define a user ID with a value of UNKNOWN.

»Managing the mqm group«
Security administrators add users who need to administer WebSphere MQ to the mqm group. This includes the root user on UNIX systems. They might also need to remove users who no longer need this authority.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:53

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12740_

6.3.2.1. Managing the mqm group

Security administrators add users who need to administer WebSphere® MQ to the mqm group. This includes the root user on UNIX systems. They might also need to remove users who no longer need this authority.

These tasks are described in Creating and managing groups.

If your domain controller runs on Windows 2000 or Windows 2003, your domain administrator might have to set up a special account for WebSphere MQ to use. This is described in the Configuring WebSphere MQ accounts.

Parent topic: »Authority to administer WebSphere MQ on UNIX and Windows systems«

This build: January 26, 2011 11:54:53

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12750_

6.3.3. Authority to work with WebSphere MQ objects on UNIX systems and Windows

All objects are protected by WebSphere® MQ, and principals must be given appropriate authority to access them. Different principals need different access rights to different objects.

Queue managers, queues, process definitions, namelists, channels, client connection channels, listeners, services, and authentication information objects are all accessed from applications that use MQI calls or PCF commands. These resources are all protected by WebSphere MQ, and applications need to be given permission to access them. The entity making the request might be a user, an application program that issues an MQI call, or an administration program that issues a PCF command. The identifier of the requester is referred to as the *principal*.

Different groups of principals can be granted different types of access authority to the same object. For example, for a specific queue, one group might be allowed to perform both put and get operations; another group might be allowed only to browse the queue (MQGET with browse option). Similarly, some groups might have put and get authority to a queue, but not be allowed to alter attributes of the queue or delete it.

Some operations are particularly sensitive and should be limited to privileged users. For example:

- Accessing some special queues, such as transmission queues or the command queue SYSTEM.ADMIN.COMMAND.QUEUE
- Running programs that use full MQI context options
- Creating and deleting application queues

Full access permission to an object is automatically given to the user ID that created the object and to all members of the mqm group (and to the members of the local Administrators group on Windows systems).

»When security checks are made on UNIX systems and Windows«
Security checks are typically made on connecting to a queue manager, opening or closing objects, and putting or getting messages.

»How access control is implemented by WebSphere MQ on UNIX systems and Windows«
WebSphere MQ uses the security services provided by the underlying operating system, using the Object Authority Manager. WebSphere MQ supplies commands to create and maintain access control lists.

»Identifying the user ID on UNIX systems and Windows«
The Object Authority Manager identifies the principal that is requesting access to a resource. The user ID used as the principal varies according to context.

»Alternate-user authority on UNIX systems and Windows«
You can specify that a user ID can use the authority of another user when accessing a WebSphere MQ object. This is called *alternate-user authority*, and you can use it on any WebSphere MQ object.

»Context authority on UNIX systems and Windows«

Context is information that applies to a particular message and is contained in the message descriptor, MQMD, which is part of the message. Applications can specify the context data when either an MQOPEN or MQPUT call is made.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12760_

6.3.3.1. When security checks are made on UNIX systems and Windows

Security checks are typically made on connecting to a queue manager, opening or closing objects, and putting or getting messages.

The security checks made for a typical application are as follows:

Connecting to the queue manager (MQCONN or MQCONNX calls)

This is the first time that the application is associated with a particular queue manager. The queue manager interrogates the operating environment to discover the user ID associated with the application. WebSphere® MQ then verifies that the user ID is authorized to connect to the queue manager and retains the user ID for future checks.

Users do not have to sign on to WebSphere MQ; WebSphere MQ assumes that users have signed on to the underlying operating system and have been authenticated by that.

Opening the object (MQOPEN or MQPUT1 calls)

WebSphere MQ objects are accessed by opening the object and issuing commands against it. All resource checks are performed when the object is opened, rather than when it is actually accessed. This means that the MQOPEN request must specify the type of access required (for example, whether the user wants only to browse the object or perform an update like putting messages onto a queue).

WebSphere MQ checks the resource that is named in the MQOPEN request. For an alias or remote queue object, the authorization used is that of the object itself, not the queue to which the alias or remote queue resolves. This means that the user does not need permission to access it. Limit the authority to create queues to privileged users. If you do not, users might bypass the normal access control simply by creating an alias. If a remote queue is referred to explicitly with both the queue and queue manager names, the transmission queue associated with the remote queue manager is checked.

The authority to a dynamic queue is based on that of the model queue from which it is derived, but is not necessarily the same. This is described in Note 1.

The user ID used by the queue manager for access checks is the user ID obtained from the operating environment of the application connected to the queue manager. A suitably authorized application can issue an MQOPEN call specifying an alternative user ID; access control checks are then made on the alternative user ID. This does not change the user ID associated with the application, only that used for access control checks.

Putting and getting messages (MQPUT or MQGET calls)

No access control checks are performed.

Closing the object (MQCLOSE)

No access control checks are performed, unless the MQCLOSE will result in a dynamic queue being deleted. In this case, there is a check that the user ID is authorized to delete the queue.

Parent topic: »Authority to work with WebSphere MQ objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12770_

6.3.3.2. How access control is implemented by WebSphere MQ on

UNIX systems and Windows

WebSphere® MQ uses the security services provided by the underlying operating system, using the Object Authority Manager. WebSphere MQ supplies commands to create and maintain access control lists.

An access control interface called the Authorization Service Interface is part of WebSphere MQ. WebSphere MQ supplies an implementation of an access control manager (conforming to the Authorization Service Interface) known as the *Object Authority Manager (OAM)*. This is automatically installed and enabled for each queue manager you create, unless you specify otherwise (as described in Preventing security access checks on UNIX systems and Windows). The OAM can be replaced by any user or vendor written component that conforms to the Authorization Service Interface.

The OAM exploits the security features of the underlying operating system, using operating system user and group IDs. Users can access WebSphere MQ objects only if they have the correct authority. Using the OAM to control access to objects on UNIX systems and Windows describes how to grant and revoke this authority.

The OAM maintains an access control list (ACL) for each resource that it controls. Authorization data is stored on a local queue called SYSTEM.AUTH.DATA.QUEUE. Access to this queue is restricted to users in the mqm group, and additionally on Windows, to users in the Administrators group, and users logged in with the SYSTEM ID. User access to the queue cannot be changed.

WebSphere MQ supplies commands to create and maintain access control lists. For more information on these commands, see Using the OAM to control access to objects on UNIX systems and Windows.

WebSphere MQ passes the OAM a request containing a principal, a resource name, and an access type. The OAM grants or rejects access based on the ACL that it maintains. WebSphere MQ follows the decision of the OAM; if the OAM cannot make a decision, WebSphere MQ does not allow access.

Parent topic: Authority to work with WebSphere MQ objects on UNIX systems and Windows

This build: January 26, 2011 11:54:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12780_

6.3.3.3. Identifying the user ID on UNIX systems and Windows

The Object Authority Manager identifies the principal that is requesting access to a resource. The user ID used as the principal varies according to context.

The Object Authority Manager (OAM) needs to be able to identify who is requesting access to a particular resource. WebSphere® MQ uses the term *principal* to refer to this identifier. The principal is established when the application first connects to the queue manager; it is determined by the queue manager from the user ID associated with the connecting application. (If the application issues XA calls without connecting to the queue manager, then the user ID associated with the application that issues the xa_open call is used for authority checks by the queue manager.)

On UNIX systems, the authorization routines checks either the real (logged-in) user ID, or the effective user ID associated with the application. The user ID checked can be dependent on the bind type, for details see Installable services.

WebSphere MQ propagates the user ID received from the system in the message header (MQMD structure) of each message as identification of the user. This identifier is part of the message context information and is described in Context authority on UNIX systems and Windows. Applications cannot alter this information unless they have been authorized to change context information.

Principals and groups
Principals can belong to groups. You can grant access to a particular resource to groups rather than to individuals, to reduce the amount of administration required. On UNIX and Linux systems all ACLs are based on groups, but on Windows systems, ACLS are based on user IDs and groups.

Windows security identifiers (SIDs)
WebSphere MQ on Windows uses the SID where it is available. If a Windows SID is not

supplied with an authorization request, WebSphere MQ identifies the user based on the user name alone, but this might result in the wrong authority being granted.

Parent topic: »Authority to work with WebSphere MQ objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12790_

6.3.3.3.1. Principals and groups

Principals can belong to groups. You can grant access to a particular resource to groups rather than to individuals, to reduce the amount of administration required. On UNIX and Linux systems all ACLs are based on groups, but on Windows systems, ACLS are based on user IDs and groups.

»For example, you might define a group consisting of users who want to run a particular application. Other users can be given access to all the resources they require by adding their user ID to the appropriate group. This process is described in Creating and managing groups.«

A principal can belong to more than one group (its group set). It has the aggregate of all the authorities granted to each group in its group set. These authorities are cached, so any changes you make to the group membership of the principal are not recognized until the queue manager is restarted, unless you issue the MQSC command REFRESH SECURITY (or the PCF equivalent).

UNIX and Linux systems

All ACLs are based on groups. When a user is granted access to a particular resource, the primary group of the user ID is included in the ACL. The individual user ID is not included and authority is granted to all members of that group. Because of this, be aware that you can inadvertently change the authority of a principal by changing the authority of another principal in the same group.

Note: To add a user to an ACL or any group, WebSphere® MQ on UNIX and Linux systems requires the user ID to have a maximum length of eight characters.

All users are nominally assigned to the default user group *nobody* and by default, no authorizations are given to this group. You can change the authorization in the *nobody* group to grant access to WebSphere MQ resources to users without specific authorizations.

»Do not define a user ID with the value "UNKNOWN". The value "UNKNOWN" is used when a user ID is too long, so arbitrary user IDs would use the access authorities of UNKNOWN.«

User IDs can contain up to 12 characters and group names up to 12 characters.

Windows systems

ACLs are based on both user IDs and groups. Checks are the same as for UNIX and Linux systems except that individual user IDs can be displayed in the ACL as well. You can have different users on different domains with the same user ID. WebSphere MQ permits user IDs to be qualified by a domain name so that these users can be given different levels of access. Group names always refer to local groups, so you do not qualify them with a domain name.

User IDs can contain up to 20 characters, domain names up to 15 characters, and group names up to 64 characters.

The OAM first checks the local security database, then the database of the primary domain, and finally the database of any trusted domains. The first user ID encountered is used by the OAM for checking. Each of these user IDs might have different group memberships on a particular computer.

Some control commands (for example, crtmqm) change authorities on WebSphere MQ objects using the Object Authority Manager (OAM). The OAM searches the security databases in the order given in the preceding paragraph to determine the authority rights for a particular user ID. As a result, the authority determined by the OAM might override the fact that a user ID is a member of the local mqm group. For example, if you issue the crtmqm command from a user ID authenticated by a domain controller that has membership of the local mqm group through a global group, the command fails if the system has a local user of the same name who is not in the local mqm group.

Parent topic: »Identifying the user ID on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12800_

6.3.3.3.2. Windows security identifiers (SIDs)

WebSphere® MQ on Windows uses the SID where it is available. If a Windows SID is not supplied with an authorization request, WebSphere MQ identifies the user based on the user name alone, but this might result in the wrong authority being granted.

On Windows systems, the security identifier (SID) is used to supplement the user ID. The SID contains information that identifies the full user account details on the Windows security account manager (SAM) database where the user is defined. When a message is created on WebSphere MQ for Windows, WebSphere MQ stores the SID in the message descriptor. When WebSphere MQ on Windows performs authorization checks, it uses the SID to query the full information from the SAM database. (The SAM database in which the user is defined must be accessible for this query to succeed.)

By default, if a Windows SID is not supplied with an authorization request, WebSphere MQ identifies the user based on the user name alone. It does this by searching the security databases in the following order:

1. The local security database
2. The security database of the primary domain
3. The security database of trusted domains

If the user name is not unique, incorrect WebSphere MQ authority might be granted. To prevent this problem, include an SID in each authorization request; the SID is used by WebSphere MQ to establish user credentials.

To specify that all authorization requests must include an SID, use regedit. Set the SecurityPolicy to NTSIDsRequired.

Parent topic: »Identifying the user ID on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12810_

6.3.3.4. Alternate-user authority on UNIX systems and Windows

You can specify that a user ID can use the authority of another user when accessing a WebSphere® MQ object. This is called *alternate-user authority*, and you can use it on any WebSphere MQ object.

Alternate-user authority is essential where a server receives requests from a program and wants to ensure that the program has the required authority for the request. The server might have the required authority, but it needs to know whether the program has the authority for the actions it has requested.

For example, assume that a server program running under user ID PAYSERV retrieves a request message from a queue that was put on the queue by user ID USER1. When the server program gets the request message, it processes the request and puts the reply back into the reply-to queue specified with the request message. Instead of using its own user ID (PAYSERV) to authorize opening the reply-to queue, the server can specify a different user ID, in this case, USER1. In this example, you can use alternate-user authority to control whether PAYSERV is allowed to specify USER1 as an alternate-user ID when it opens the reply-to queue.

The alternate-user ID is specified on the AlternateUserId field of the object descriptor.

Parent topic: »Authority to work with WebSphere MQ objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa12820_

6.3.3.5. Context authority on UNIX systems and Windows

Context is information that applies to a particular message and is contained in the message descriptor, MQMD, which is part of the message. Applications can specify the context data when either an MQOPEN or MQPUT call is made.

The context information comes in two sections:

Identity section

Who the message came from. It consists of the UserIdentifier, AccountingToken, and ApplIdentityData fields.

Origin section

Where the message came from, and when it was put onto the queue. It consists of the PutApplType, PutApplName, PutDate, PutTime, and ApplOriginData fields.

Applications can specify the context data when either an MQOPEN or MQPUT call is made. This data might be generated by the application, passed on from another message, or generated by the queue manager by default. For example, context data can be used by server programs to check the identity of the requester, testing whether the message came from an application running under an authorized user ID.

A server program can use the UserIdentifier to determine the user ID of an alternate user. You use context authorization to control whether the user can specify any of the context options on any MQOPEN or MQPUT1 call.

See the »Controlling context information« for information about the context options, and the WebSphere MQ Application Programming Reference for descriptions of the message descriptor fields relating to context.

Parent topic: »Authority to work with WebSphere MQ objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12830_

6.3.4. Connecting to WebSphere MQ using Terminal Services

The "Create global objects" user right can cause problems if you are using Terminal Services.

If you are connecting to a Windows system by using Terminal Services and you have problems creating or starting a queue manager, this might be because of the introduction of the user right, "Create global objects", in recent versions of Windows.

The "Create global objects" user right limits the users authorized to create objects in the global namespace. In order for an application to create a global object, it must either be running in the global namespace, or the user under which the application is running must have the "Create global objects" user right applied to it.

»Administrators have the "Create global objects" user right applied by default, so an administrator can create and start queue managers when connected by using Terminal Services without altering the user rights.«

If the various methods of administering WebSphere® MQ do no work when you use terminal services, try setting the "Create global objects" user right:

1. Open the Administrative Tools panel:

   Windows 2003 and Windows XP

   Access this panel using Control Panel > Administrative Tools.

   Windows Vista and Windows Server 2008

   Access this panel using Control Panel > System and Maintenance > Administrative Tools.
2. Double-click Local Security Policy.
3. Expand Local Policies.
4. Click User Rights Assignment.
5. Add the new user or group to the "Create global objects" policy.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12840_

6.3.5. Configuring additional authority for Windows applications connecting to WebSphere MQ

The account under which WebSphere® MQ processes run might need additional authorization before SYNCHRONIZE access to application processes can be granted.

You might experience problems if you have Windows applications, for example ASP pages, connecting to WebSphere MQ that are configured to run at a security level higher than usual.

WebSphere MQ requires SYNCHRONIZE access to application processes in order to coordinate certain actions. APAR IC35116 changed WebSphere MQ so that the appropriate privileges are specified. However, the account under which WebSphere MQ processes run might need additional authorization before the requested access can be granted. Directions for configuring additional authority are given below.

To configure additional authority to the user ID under which WebSphere MQ processes are running:

- »Start the Local Security Policy tool, click Security Settings ->Local Policies->User Right Assignments, click "Debug Programs".«
- Double click "Debug Programs", then add your WebSphere MQ user ID to the list

If the system is in a Windows domain and the effective policy setting is still not set, even though the local policy setting is set, the user ID must be authorized in the same way at domain level, using the Domain Security Policy tool.
Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12850_

6.3.6. Creating and managing groups

This section tells you how to create groups and add users to them. It also describes how to remove a user from a group. Any changes you make to a principal's group membership are not recognized until the queue manager is restarted, unless you issue the MQSC command REFRESH SECURITY (or the PCF equivalent).

»Creating and managing groups on Windows«
These instructions lead you through the process of administering groups on a workstation or member server machine.

»Creating and managing groups on HP-UX«
On HP-UX, providing you are not using NIS or NIS+, use the System Administration Manager (SAM) to work with groups.

»Creating and managing groups on AIX«
On AIX®, providing you are not using NIS or NIS+, use SMITTY to work with groups.

»Creating and managing groups on Solaris«
On Solaris, providing you are not using NIS or NIS+, use the /etc/group file to work with groups.

»Creating and managing groups on Linux«
On Linux, providing you are not using NIS or NIS+, use the /etc/group file to work with groups.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12860_

6.3.6.1. Creating and managing groups on Windows

These instructions lead you through the process of administering groups on a workstation or member server machine.

For domain controllers, users and groups are administered through Active Directory. For more details on using Active Directory refer to the appropriate operating system instructions.

Any changes you make to a principal's group membership are not recognized until the queue manager is restarted, or you issue the MQSC command REFRESH SECURITY (or the PCF equivalent).

Use the Computer Management panel to work with user and groups.

Windows 2003 and Windows XP

Access this panel using Control Panel > Administrative Tools > Computer Management.

Windows Vista and Windows Server 2008

Access this panel using Control Panel > System and Maintenance > Administrative Tools > Computer Management.

»Creating a group on Windows«
Create a group by using the control panel.

»Adding a user to a group on Windows«
Add a user to a group by using the control panel

»Displaying who is in a group on Windows«
Display the members of a group by using the control panel.

»Removing a user from a group on Windows«
Remove a user from a group by using the control panel.

Parent topic: »Creating and managing groups«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12910_

6.3.6.1.1. Creating a group on Windows

Create a group by using the control panel.

Procedure

1. Open the control panel
2. Double-click Administrative Tools. The Administrative Tools panel opens.
3. Double-click Computer Management. The Computer Management panel opens.
4. Expand Local Users and Groups.
5. Right-click Groups, and select New Group.... The New Group panel is displayed.
6. Type an appropriate name in the Group name field, then click Create.
7. Click Close.

Parent topic: »Creating and managing groups on Windows«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa12920_

6.3.6.1.2. Adding a user to a group on Windows

Add a user to a group by using the control panel

Procedure

1. Open the control panel
2. Double-click Administrative Tools. The Administrative Tools panel opens.
3. Double-click Computer Management. The Computer Management panel opens.
4. From the Computer Management panel, expand Local Users and Groups.
5. Select Users
6. Double-click the user that you want to add to a group. The user properties panel is displayed.
7. Select the Member Of tab.
8. Select the group that you want to add the user to. If the group you want is not visible:
   a. Click Add.... The Select Groups panel is displayed.
   b. Click Locations.... The Locations panel is displayed.
   c. Select the location of the group you want to add the user to from the list and click OK.
   d. Type the group name in the field provided.
      Alternatively, click Advanced... and then Find Now to list the groups available in the currently selected location. From here, select the group you want to add the user to and click OK.
   e. Click OK. The user properties panel is displayed, showing the group you added.
   f. Select the group.
9. Click OK. The Computer Management panel is displayed.

Parent topic: »Creating and managing groups on Windows«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12930

6.3.6.1.3. Displaying who is in a group on Windows

Display the members of a group by using the control panel.

Procedure

1. Open the control panel
2. Double-click Administrative Tools. The Administrative Tools panel opens.
3. Double-click Computer Management. The Computer Management panel opens.
4. From the Computer Management panel, expand Local Users and Groups.
5. Select Groups.
6. Double-click a group. The group properties panel is displayed. The group properties panel is displayed.

Results

The group members are displayed.

Parent topic: »Creating and managing groups on Windows«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12940_

6.3.6.1.4. Removing a user from a group on Windows

Remove a user from a group by using the control panel.

Procedure

1. Open the control panel
2. Double-click Administrative Tools. The Administrative Tools panel opens.
3. Double-click Computer Management. The Computer Management panel opens.
4. From the Computer Management panel, expand Local Users and Groups.
5. Select Users.
6. Double-click the user that you want to add to a group. The user properties panel is displayed.
7. Select the Member Of tab.
8. Select the group that you want to remove the user from, then click Remove.
9. Click OK. The Computer Management panel is displayed.

Results

You have now removed the user from the group.

Parent topic: »Creating and managing groups on Windows«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12950_

6.3.6.2. Creating and managing groups on HP-UX

On HP-UX, providing you are not using NIS or NIS+, use the System Administration Manager (SAM) to work with groups.

»Creating a group on HP-UX«
Add a user to a group by using the System Administration Manager

»Adding a user to a group on HP-UX«
Add a user to a group by using the System Administration Manager

»Displaying who is in a group on HP-UX«
Display who is in a group by using the System Administration Manager

»Removing a user from a group on HP-UX«
Remove a user from a group by using the System Administration Manager.

Parent topic: »Creating and managing groups«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12960_

6.3.6.2.1. Creating a group on HP-UX

Add a user to a group by using the System Administration Manager

Procedure

1. From the System Administration Manager (SAM), double click Accounts for Users and Groups.
2. Double click Groups.
3. Select Add from the Actions pull down to display the Add a New Group panel.
4. Enter the name of the group and select the users that you want to add to the group.
5. Click Apply to create the group.

Results

You have now created a group.

Parent topic: »Creating and managing groups on HP-UX«

This build: January 26, 2011 11:54:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12970_

6.3.6.2.2. Adding a user to a group on HP-UX

Add a user to a group by using the System Administration Manager

Procedure

1. From the System Administration Manager (SAM), double click Accounts for Users and Groups.
2. Double click Groups.
3. Highlight the name of the group and select Modify from the Actions pull down to display the Modify an Existing Group panel.
4. Select a user that you want to add to the group and click Add.
5. If you want to add other users to the group, repeat step 4 for each user.
6. When you have finished adding names to the list, click OK.

Results

You have now added a user to a group.

Parent topic: »Creating and managing groups on HP-UX«

This build: January 26, 2011 11:54:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12980_

6.3.6.2.3. Displaying who is in a group on HP-UX

Display who is in a group by using the System Administration Manager

Procedure

1. From the System Administration Manager (SAM), double click Accounts for Users and Groups.
2. Double click Groups.
3. Highlight the name of the group and select Modify from the Actions pull down to display the Modify an Existing Group panel, showing a list of the users in the group.

Results

The group members are displayed.

Parent topic: »Creating and managing groups on HP-UX«

This build: January 26, 2011 11:54:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa12990_

6.3.6.2.4. Removing a user from a group on HP-UX

Remove a user from a group by using the System Administration Manager.

Procedure

1. From the System Administration Manager (SAM), double click Accounts for Users and Groups.

2. Double click Groups.
3. Highlight the name of the group and select Modify from the Actions pull down to display the Modify an Existing Group panel.
4. Select a user that you want to remove from the group and click Remove.
5. If you want to remove other users from the group, repeat step 4 for each user.
6. When you have finished removing names from the list, click OK.

Results

You have now removed a user from a group

Parent topic: »Creating and managing groups on HP-UX«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13000_

6.3.6.3. Creating and managing groups on AIX

On AIX®, providing you are not using NIS or NIS+, use SMITTY to work with groups.

»Creating a group«
Create a group using SMITTY.

»Adding a user to a group«
Add a user to a group by using SMITTY

»Displaying who is in a group«
Display who is in a group using SMITTY.

»Removing a user from a group«
Remove a user from a group by using SMITTY

Parent topic: »Creating and managing groups«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13010_

6.3.6.3.1. Creating a group

Create a group using SMITTY.

Procedure

1. From SMITTY, select Security and Users and press Enter.
2. Select Groups and press Enter.
3. Select Add a Group and press Enter.
4. Enter the name of the group and the names of any users that you want to add to the group, separated by commas.
5. Press Enter to create the group.

Results

You have now created a group.

Parent topic: »Creating and managing groups on AIX«

This build: January 26, 2011 11:54:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa13020_

6.3.6.3.2. Adding a user to a group

Add a user to a group by using SMITTY

Procedure

1. From SMITTY, select Security and Users and press Enter.
2. Select Groups and press Enter.
3. Select Change / Show Characteristics of Groups and press Enter.
4. Enter the name of the group to show a list of the members of the group.
5. Add the names of the users that you want to add to the group, separated by commas.
6. Press Enter to add the names to the group.

Parent topic: »Creating and managing groups on AIX«

This build: January 26, 2011 11:54:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13030_

6.3.6.3.3. Displaying who is in a group

Display who is in a group using SMITTY.

Procedure

1. From SMITTY, select Security and Users and press Enter.
2. Select Groups and press Enter.
3. Select Change / Show Characteristics of Groups and press Enter.
4. Enter the name of the group to show a list of the members of the group.

Results

The group members are displayed.

Parent topic: »Creating and managing groups on AIX«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13040_

6.3.6.3.4. Removing a user from a group

Remove a user from a group by using SMITTY

Procedure

1. From SMITTY, select Security and Users and press Enter.
2. Select Groups and press Enter.
3. Select Change / Show Characteristics of Groups and press Enter.
4. Enter the name of the group to show a list of the members of the group.
5. Delete the names of the users that you want to remove from the group.
6. Press Enter to remove the names from the group.

Results

You have now removed a user from a group.

Parent topic: »Creating and managing groups on AIX«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13050_

6.3.6.4. Creating and managing groups on Solaris

On Solaris, providing you are not using NIS or NIS+, use the /etc/group file to work with groups.

»Creating a group on Solaris«
Creating a group by using the groupadd command.

»Adding a user to a group on Solaris«
Add a user to a group by using the usermod command.

»Displaying who is in a group on Solaris«
To discover who is a member of a group, look at the entry for that group in the /etc/group file.

»Removing a user from a group on Solaris«
Remove a user from a group by using the usermod command

Parent topic: »Creating and managing groups«

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13060_

6.3.6.4.1. Creating a group on Solaris

Creating a group by using the groupadd command.

Procedure

Type the following command: groupadd *group-name* where *group-name* is the name of the group.

Results

The file /etc/group file holds group information.

Parent topic: »Creating and managing groups on Solaris«

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13070_

6.3.6.4.2. Adding a user to a group on Solaris

Add a user to a group by using the usermod command.

Procedure

To add a member to a supplementary group, execute the usermod command and list the supplementary groups that the user is currently a member of, and the supplementary groups that the user is to become a member of. For example, if the user is a member of the group groupa, and is to become a member of groupb also, use the following command: usermod -G groupa,groupb *user-name* , where *user-name* is the user name.

Parent topic: »Creating and managing groups on Solaris«

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13080_

6.3.6.4.3. Displaying who is in a group on Solaris

To discover who is a member of a group, look at the entry for that group in the /etc/group file.

Parent topic: »Creating and managing groups on Solaris«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13090_

6.3.6.4.4. Removing a user from a group on Solaris

Remove a user from a group by using the usermod command

Procedure

To remove a member from a supplementary group, execute the usermod command listing the supplementary groups that you want the user to remain a member of. For example, if the user's primary group is users and the user is also a member of the groups mqm, groupa and groupb, to remove the user from the mqm group, the following command is used: usermod -G groupa,groupb *user-name*, where *user-name* is the user name.

Parent topic: »Creating and managing groups on Solaris«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13100_

6.3.6.5. Creating and managing groups on Linux

On Linux, providing you are not using NIS or NIS+, use the /etc/group file to work with groups.

»Creating a group on Linux«
Create a group by using the Groupadd command.

»Adding a user to a group on Linux«
Add a user to a group by using the usermod command.

»Displaying who is in a group on Linux«
Display who is in a group by using the getent command.

»Removing a user from a group«
Remove a user from a group by using the usermod command.

Parent topic: »Creating and managing groups«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13110_

6.3.6.5.1. Creating a group on Linux

Create a group by using the Groupadd command.

Procedure

To create a new group, type the following command: Groupadd -g *group-ID group-name* , where *group-ID* is the numeric identifier of the group, and *group-name* is the name of the group.

Results

The file /etc/group file holds group information.

Parent topic: »Creating and managing groups on Linux«

This build: January 26, 2011 11:54:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13120_

6.3.6.5.2. Adding a user to a group on Linux

Add a user to a group by using the usermod command.

Procedure

To add a member to a supplementary group, execute the usermod command and list the supplementary groups that the user is currently a member of, and the supplementary groups that the user is to become a member of. For example, if the user is a member of the group groupa, and is to become a member of groupb also, the following command is used: usermod -G groupa,groupb *user-name* , where *user-name* is the user name.
Parent topic: »Creating and managing groups on Linux«

This build: January 26, 2011 11:54:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13130_

6.3.6.5.3. Displaying who is in a group on Linux

Display who is in a group by using the getent command.

Procedure

To display who is a member of a group, type the following command: getent group *group-name* , where *group-name* is the name of the group.
Parent topic: »Creating and managing groups on Linux«

This build: January 26, 2011 11:54:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13140_

6.3.6.5.4. Removing a user from a group

Remove a user from a group by using the usermod command.

Procedure

To remove a member from a supplementary group, execute the usermod command listing the supplementary groups that you want the user to remain a member of. For example, if the user's primary group is users and the user is also a member of the groups mqm, groupa and groupb, to remove the user from the mqm group, the following command is used: usermod -G groupa,groupb *user-name* , where *user-name* is the user name.
Parent topic: »Creating and managing groups on Linux«

This build: January 26, 2011 11:54:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13150_

6.3.7. Using the OAM to control access to objects on UNIX systems and Windows

The object authority manager (OAM) provides a command interface for granting and revoking authority to WebSphere® MQ objects.

You must be suitably authorized to use these commands, as described in Authority to administer WebSphere MQ on UNIX and Windows systems. User IDs that are authorized to

administer WebSphere MQ have *super user* authority to the queue manager, which means that you do not have to grant them further permission to issue any MQI requests or commands.

»Giving access to a WebSphere MQ object on UNIX systems and Windows«
Use the setmqaut control command, or the MQCMD_SET_AUTH_REC PCF command to give users, and groups of users, access to WebSphere MQ objects.

»Using OAM generic profiles on UNIX systems and Windows«
OAM generic profiles enable you to set the authority a user has to many objects at once, rather than having to issue separate setmqaut commands against each individual object when it is created.

»Displaying access settings«
Use the dspmqaut control command, or the MQCMD_INQUIRE_ENTITY_AUTH PCF command to view the authorizations that a specific principal or group has for a particular object.

»Changing and revoking access to a WebSphere MQ object«
To change the level of access that a user or group has to an object, use the setmqaut command. To revoke the access of a particular user that is a member of a group that has authorization, remove the user from the group.

»Preventing security access checks on UNIX systems and Windows«
To turn off all security checking you can disable the OAM. This might be suitable for a test environment. Having disabled or removed the OAM, you cannot add an OAM to an existing queue manager.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13160_

6.3.7.1. Giving access to a WebSphere MQ object on UNIX systems and Windows

Use the setmqaut control command, or the MQCMD_SET_AUTH_REC PCF command to give users, and groups of users, access to WebSphere® MQ objects.

For a full definition of the setmqaut control command and its syntax, see setmqaut, and for a full definition of the MQCMD_SET_AUTH_REC PCF command and its syntax, see Set Authority Record.

The queue manager must be running to use this command. When you have changed access for a principal, the changes are reflected immediately by the OAM.

To give users access to an object, you need to specify:

- The name of the queue manager that owns the objects you are working with; if you do not specify the name of a queue manager, the default queue manager is assumed.
- The name and type of the object (to identify the object uniquely). You specify the name as a *profile*; this is either the explicit name of the object, or a generic name, including wildcard characters. For a detailed description of generic profiles, and the use of wildcard characters within them, see Using OAM generic profiles on UNIX systems and Windows.
- One or more principals and group names to which the authority applies.
  If a user ID contains spaces, enclose it in quotation marks when you use this command. On Windows systems, you can qualify a user ID with a domain name. If the actual user ID contains an at sign (@) symbol, replace it with @@ to show that it is part of the user ID, not the delimiter between the user ID and the domain name.
- A list of authorizations. Each item in the list specifies a type of access that is to be granted to that object (or revoked from it). Each authorization in the list is specified as a keyword, prefixed with a plus sign (+) or a minus sign (-). Use a plus sign to add the specified authorization, and a minus sign to remove the authorization. There must be no spaces between the + or - sign and the keyword.
  You can specify any number of authorizations in a single command. For example, the list of authorizations to permit a user or group to put messages on a queue and to browse them, but to revoke access to get messages is:

```
+browse -get +put
```

Using the command with a different authorization service

If you are using your own authorization service instead of the OAM, you can specify the name of this service on the setmqaut command to direct the command to this service. You must specify this parameter if you have multiple installable components running at the same time; if you do not, the update is made to the first installable component for the authorization service. By default, this is the supplied OAM.

Examples of using the setmqaut command
Examples of granting and revoking permissions to use an object, and revoking put authority.

Using the command with a different authorization service

Parent topic: Using the OAM to control access to objects on UNIX systems and Windows

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13170_

6.3.7.1.1. Examples of using the setmqaut command

Examples of granting and revoking permissions to use an object, and revoking put authority.

The following examples show how to use the setmqaut command to grant and revoke permission to use an object:

```
setmqaut -m saturn.queue.manager -t queue -n RED.LOCAL.QUEUE
         -g groupa +browse -get +put
```

In this example:

- saturn.queue.manager is the queue manager name
- queue is the object type
- RED.LOCAL.QUEUE is the object name
- groupa is the identifier of the group whose authorizations are to change
- +browse -get +put is the authorization list for the specified queue
  - +browse adds authorization to browse messages on the queue (to issue MQGET with the browse option)
  - -get removes authorization to get (MQGET) messages from the queue
  - +put adds authorization to put (MQPUT) messages on the queue

The following command revokes put authority on the queue MyQueue from principal fvuser and from groups groupa and groupb. On UNIX systems, this command also revokes put authority for all principals in the same primary group as fvuser.

```
setmqaut -m saturn.queue.manager -t queue -n MyQueue -p fvuser
         -g groupa -g groupb -put
```

Parent topic: Giving access to a WebSphere MQ object on UNIX systems and Windows

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13180_

6.3.7.1.2. Using the command with a different authorization service

If you are using your own authorization service instead of the OAM, you can specify the name of this service on the setmqaut command to direct the command to this service. You must specify this parameter if you have multiple installable components running at the same time; if you do not, the update is made to the first installable component for the authorization

service. By default, this is the supplied OAM.

Parent topic: »Giving access to a WebSphere MQ object on UNIX systems and Windows«

This build: January 26, 2011 11:54:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13190_

6.3.7.2. Using OAM generic profiles on UNIX systems and Windows

OAM generic profiles enable you to set the authority a user has to many objects at once, rather than having to issue separate setmqaut commands against each individual object when it is created.

Using generic profiles in the setmqaut command enables you to set a generic authority for all objects that fit that profile.

This collection of topics describes the use of generic profiles in more detail.

»Using wildcard characters in OAM profiles«
Use wildcard characters in an object authority manager (OAM) profile name to make that profile applicable to more than one object.

»Profile priorities«
More than one generic profile can apply to a single object. Where this is the case, the most specific rule applies.

»Dumping profile settings«
Use the dmpmqaut control command or the MQCMD_INQUIRE_AUTH_RECS PCF command to dump the current authorizations associated with a specified profile.

Parent topic: »Using the OAM to control access to objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13200_

6.3.7.2.1. Using wildcard characters in OAM profiles

Use wildcard characters in an object authority manager (OAM) profile name to make that profile applicable to more than one object.

What makes a profile generic is the use of special characters (wildcard characters) in the profile name. For example, the question mark (?) wildcard character matches any single character in a name. So, if you specify ABC.?EF, the authorization you give to that profile applies to any objects with the names ABC.DEF, ABC.CEF, ABC.BEF, and so on.

The wildcard characters available are:

?

Use the question mark (?) instead of any single character. For example, AB.?D applies to the objects AB.CD, AB.ED, and AB.FD.

*

Use the asterisk (*) as:

- A *qualifier* in a profile name to match any one qualifier in an object name. A qualifier is the part of an object name delimited by a period. For example, in ABC.DEF.GHI, the qualifiers are ABC, DEF, and GHI.
  For example, ABC.*.JKL applies to the objects ABC.DEF.JKL, and ABC.GHI.JKL. (Note that it does not apply to ABC.JKL; * used in this context always indicates one qualifier.)
- A character within a qualifier in a profile name to match zero or more characters within the qualifier in an object name.
  For example, ABC.DE*.JKL applies to the objects ABC.DE.JKL, ABC.DEF.JKL, and ABC.DEGH.JKL.

**

Use the double asterisk (**) once in a profile name as:

- The entire profile name to match all object names. For example if you use -t prcs to identify processes, then use ** as the profile name, you change the authorizations for all processes.
- As either the beginning, middle, or ending qualifier in a profile name to match zero or more qualifiers in an object name. For example, **.ABC identifies all objects with the final qualifier ABC.

Note: When using wildcard characters on UNIX systems, you must enclose the profile name in quotes.

Parent topic: »Using OAM generic profiles on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13210_

6.3.7.2.2. Profile priorities

More than one generic profile can apply to a single object. Where this is the case, the most specific rule applies.

An important point to understand when using generic profiles is the priority that profiles are given when deciding what authorities to apply to an object being created. For example, suppose that you have issued the commands:

```
setmqaut -n AB.* -t q +put -p fred
setmqaut -n AB.C* -t q +get -p fred
```

The first gives put authority to all queues for the principal fred with names that match the profile AB.*; the second gives get authority to the same types of queue that match the profile AB.C*.

Suppose that you now create a queue called AB.CD. According to the rules for wildcard matching, either setmqaut could apply to that queue. So, does it have put or get authority?

To find the answer, you apply the rule that, whenever multiple profiles can apply to an object, only the most specific applies. The way that you apply this rule is by comparing the profile names from left to right. Wherever they differ, a non-generic character is more specific then a generic character. So, in the example above, the queue AB.CD has get authority (AB.C* is more specific than AB.*).

When you are comparing generic characters, the order of *specificity* is:

1. ?
2. *
3. **

Parent topic: »Using OAM generic profiles on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13220_

6.3.7.2.3. Dumping profile settings

Use the dmpmqaut control command or the MQCMD_INQUIRE_AUTH_RECS PCF command to dump the current authorizations associated with a specified profile.

For a full definition of the dmpmqaut control command and its syntax, see dmpmqaut, and for a full definition of the MQCMD_INQUIRE_AUTH_RECS PCF command and its syntax, see Inquire Authority Records book.

The following examples show the use of the dmpmqaut control command to dump authority records for generic profiles:

1. This example dumps all authority records with a profile that matches queue a.b.c for principal user1.

   ```
   dmpmqaut -m qm1 -n a.b.c -t q -p user1
   ```

The resulting dump looks something like this:

```
profile:      a.b.*
object type: queue
entity:       user1
type:         principal
authority:    get, browse, put, inq
```

Note: UNIX users cannot use the -p option; they must use -g groupname instead.

2. This example dumps all authority records with a profile that matches queue a.b.c.

```
dmpmqaut -m qmgr1 -n a.b.c -t q
```

The resulting dump looks something like this:

```
profile:      a.b.c
object type: queue
entity:       Administrator
type:         principal
authority:    all
- - - - - - - - - - - - - - - - -
profile:      a.b.*
object type: queue
entity:       user1
type:         principal
authority:    get, browse, put, inq
- - - - - - - - - - - - - - - - -
profile:      a.**
object type: queue
entity:       group1
type:         group
authority:    get
```

3. This example dumps all authority records for profile a.b.*, of type queue.

```
dmpmqaut -m qmgr1 -n a.b.* -t q
```

The resulting dump looks something like this:

```
profile:      a.b.*
object type: queue
entity:       user1
type:         principal
authority:    get, browse, put, inq
```

4. This example dumps all authority records for queue manager qmX.

```
dmpmqaut -m qmX
```

The resulting dump looks something like this:

```
profile:      q1
object type: queue
entity:       Administrator
type:         principal
authority:    all
- - - - - - - - - - - - - - - - -
profile:      q*
object type: queue
entity:       user1
type:         principal
authority:    get, browse
- - - - - - - - - - - - - - - - -
profile:      name.*
object type: namelist
entity:       user2
type:         principal
authority:    get
- - - - - - - - - - - - - - - - -
profile:      pr1
object type: process
entity:       group1
type:         group
authority:    get
```

5. This example dumps all profile names and object types for queue manager qmX.

```
dmpmqaut -m qmX -l
```

The resulting dump looks something like this:

```
profile: q1, type: queue
profile: q*, type: queue
```

```
profile: name.*, type: namelist
profile: pr1, type: process
```

Note: For WebSphere® MQ for Windows only, all principals displayed include domain information, for example:

```
profile:      a.b.*
object type:  queue
entity:       user1@domain1
type:         principal
authority:    get, browse, put, inq
```

Parent topic: »Using OAM generic profiles on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13230_

6.3.7.3. Displaying access settings

Use the dspmqaut control command, or the MQCMD_INQUIRE_ENTITY_AUTH PCF command to view the authorizations that a specific principal or group has for a particular object.

The queue manager must be running to use this command. When you change access for a principal, the changes are reflected immediately by the OAM. Authorization can be displayed for only one group or principal at a time. For a full definition of the dmpmqaut control command and its syntax, see dmpmqaut, and for a full definition of the MQCMD_INQUIRE_ENTITY_AUTH PCF command and its syntax, see Inquire Entity Authority.

The following example shows the use of the dspmqaut control command to display the authorizations that the group GpAdmin has to a process definition named Annuities that is on queue manager QueueMan1.

```
dspmqaut -m QueueMan1 -t process -n Annuities -g GpAdmin
```

Parent topic: »Using the OAM to control access to objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13240_

6.3.7.4. Changing and revoking access to a WebSphere MQ object

To change the level of access that a user or group has to an object, use the setmqaut command. To revoke the access of a particular user that is a member of a group that has authorization, remove the user from the group.

The process of removing the user from a group is described in Creating and managing groups.

The user ID that creates a WebSphere® MQ object is granted full control authorities to that object. If you remove this user ID from the local mqm group (or the Administrators group on Windows systems) these authorities are not revoked. Use the setmqaut control command or the MQCMD_DELETE_AUTH_REC PCF command to revoke access to an object for the user ID that created it, after removing it from the mqm or Administrators group. For a full definition of the setmqaut control command and its syntax, see setmqaut, and for a full definition of the MQCMD_INQUIRE_ENTITY_AUTH PCF command and its syntax, see Inquire Entity Authority.

On Windows, delete the OAM entries corresponding to a particular Windows user account before deleting the user profile. It is impossible to remove the OAM entries after removing the user account.

Parent topic: »Using the OAM to control access to objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13250_

6.3.7.5. Preventing security access checks on UNIX systems and Windows

To turn off all security checking you can disable the OAM. This might be suitable for a test environment. Having disabled or removed the OAM, you cannot add an OAM to an existing queue manager.

If you decide that you do not want to perform security checks (for example, in a test environment), you can disable the OAM in one of two ways:

- Before you create a queue manager, set the operating system environment variable MQSNOAUT as follows (if you do this, you cannot add an OAM later):
  On Windows systems:

  ```
  SET MQSNOAUT=yes
  ```

  On UNIX systems:

  ```
  export MQSNOAUT=yes
  ```

  See MQSNOAUT for more information about the implications of setting the MQSNOAUT variable.
- Use the WebSphere® MQ Explorer or edit the queue manager configuration file to remove the service (if you do this, you cannot add an OAM later).

If you use setmqaut, or dspmqaut while the OAM is disabled, note the following points:

- The OAM does not validate the specified principal, or group, meaning that the command can accept invalid values.
- The OAM does not perform security checks and indicates that all principals and groups are authorized to perform all applicable object operations.

When an OAM is removed, it cannot be put back on an existing queue manager. This is because the OAM needs to be in place at object creation time. To use the WebSphere MQ OAM again after it has been removed, the queue manager needs to be rebuilt.

Parent topic: »Using the OAM to control access to objects on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13260_

6.3.8. Channel security

User IDs for sending and receiving channels need access to various WebSphere MQ resources. For receiving channels, you can choose to use the user ID associated with the MCA or the message.

Message channel agents (MCAs) are WebSphere® MQ applications and need access to various WebSphere MQ resources.

- The user ID associated with a sending channel needs access to the queue manager, the transmission queue, the dead-letter queue, and any resources required by channel exits.
- The user ID associated with the receiving channel needs to open the target queues to put messages onto them. This involves the MQI, so access control checks might need to be made. You can specify whether these checks are made against the user ID associated with the MCA (as described below), or the user ID associated with the message (from the MQMD context field).
  »For the channel types to which it applies, the PUTAUT parameter of a channel definition specifies which user ID is used for these checks.
  - »If you use the default user ID, this user ID will already be defined on the local system.«
  - If you use the user ID associated with the message, it is likely that this is a user ID from a remote system. This remote system user ID must be recognized by the target system and have the authority to connect to the queue manager, make inquiries, set attributes, and set context options (+connect, +inq, +set, and +setall). It must also have authority to put messages and set context information (+put and +setall) for the destination and dead-letter queues.

  «

The user ID associated with the MCA depends on the type of MCA.

Caller MCA

These are MCAs that initiate a channel. They can be started as individual processes, as threads of the channel initiator, or as threads of a process pool. The user ID used is that associated with the parent process (the channel initiator), or the process causing the MCA to be started.

Responder MCA

These are MCAs that are started as a result of a request by a caller MCA. They can be started as individual processes, as threads of the listener, or as threads of a process pool. The user ID can be any one of the following (in this order of preference):

1. On APPC, the caller MCA can indicate the user ID to be used for the responder MCA. This is called the network user ID and applies only to channels started as individual processes. This is set using the USERID parameter of the channel definition.
2. If the USERID parameter is not used, the channel definition of the responder MCA can specify the user ID that the MCA is to use. This is set using the MCAUSER parameter of the channel definition.
3. If the user ID has not been set by either of the methods above, the user ID of the process that starts the MCA or the user ID of the parent process (the listener) is used.

»Protecting channel initiator definitions«
Only members of the mqm group can manipulate channel initiators.

»Transmission queues«
Queue managers automatically put remote messages on a transmission queue; no special authority is required for this.

»Channel exits«
You can use channel exits for added security. A security exit forms a secure connection between two security exit programs, where one program is for the sending message channel agent (MCA), and one is for the receiving MCA.

»Protecting channels with SSL«
SSL support in WebSphere MQ uses the queue manager authentication information object and various MQSC commands and queue manager and channel parameters that define the SSL support required in detail.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13270_

6.3.8.1. Protecting channel initiator definitions

Only members of the mqm group can manipulate channel initiators.

WebSphere® MQ channel initiators are not WebSphere MQ objects; access to them is not controlled by the OAM. WebSphere MQ does not allow users or applications to manipulate these objects, unless their user ID is a member of the mqm group. If you have an application that issues the PCF command StartChannelInitiator, the user ID specified in the message descriptor of the PCF message must be a member of the mqm group on the target queue manager.

A user ID must also be a member of the mqm group on the target machine to issue the equivalent MQSC commands through the Escape PCF command or using runmqsc in indirect mode.

Parent topic: »Channel security«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13280_

6.3.8.2. Transmission queues

Queue managers automatically put remote messages on a transmission queue; no special authority is required for this.

However, if you need to put a message directly on a transmission queue, this requires special authorization; see Table 3.

Parent topic: »Channel security«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13290_

6.3.8.3. Channel exits

You can use channel exits for added security. A security exit forms a secure connection between two security exit programs, where one program is for the sending message channel agent (MCA), and one is for the receiving MCA.

See WebSphere MQ Application Programming Guide and WebSphere MQ Intercommunication for more information about channel exits.

Parent topic: »Channel security«

This build: January 26, 2011 11:54:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13300_

6.3.8.4. Protecting channels with SSL

SSL support in WebSphere® MQ uses the queue manager authentication information object and various MQSC commands and queue manager and channel parameters that define the SSL support required in detail.

The Secure Sockets Layer (SSL) protocol provides channel security, with protection against eavesdropping, tampering, and impersonation. WebSphere MQ support for SSL enables you to specify, on the channel definition, that a particular channel uses SSL security. You can also specify details of the kind of security you want, such as the encryption algorithm you want to use.

The following MQSC commands support SSL:

ALTER AUTHINFO

Modifies the attributes of an authentication information object.

DEFINE AUTHINFO

Creates a new authentication information object.

DELETE AUTHINFO

Deletes an authentication information object.

DISPLAY AUTHINFO

Displays the attributes for a specific authentication information object.

The following queue manager parameters support SSL:

SSLCRLNL

»The SSLCRLNL attribute specifies a namelist of authentication information objects which are used to provide certificate revocation locations to allow enhanced TLS/SSL certificate checking.«

SSLCRYP

On Windows and UNIX systems, sets the SSLCryptoHardware queue manager attribute. This attribute is the name of the parameter string that you can use to configure the cryptographic hardware you have on your system.

SSLEV

Determines whether an SSL event message will be reported if a channel using SSL fails to establish an SSL connection.

SSLFIPS

Specifies whether only FIPS-certified algorithms are to be used if cryptography is carried out in WebSphere MQ. If cryptographic hardware is configured, the cryptographic modules used are those provided by the hardware product, and these may, or may not, be FIPS-certified to a particular level. This depends on the hardware product in use.

SSLKEYR

On Windows and UNIX systems, associates a key repository with a queue manager. The key database is held in a *GSKit* key database. (The IBM® Global Security Kit (GSKit) enables you to use SSL security on Windows and UNIX systems systems.)

SSLRKEYC

The number of bytes to be sent and received within an SSL conversation before the secret key is renegotiated. The number of bytes includes control information sent by the MCA.

The following channel parameters support SSL:

SSLCAUTH

Defines whether WebSphere MQ requires and validates a certificate from the SSL client.

SSLCIPH

Specifies the encryption strength and function (CipherSpec), for example NULL_MD5 or RC4_MD5_US. The CipherSpec must match at both ends of channel.

SSLPEER

Specifies the distinguished name (unique identifier) of allowed partners.

This book describes the setmqaut, dspmqaut, dmpmqaut, rcrmqobj, rcdmqimg, and dspmqfls commands to support the authentication information object. It also describes the amqtcert command for migrating certificates on Windows systems, the iKeycmd command for managing certificates on UNIX systems, and the GSKCapiCmd tool for managing certificates on UNIX and Windows systems. See the following sections:

- setmqaut
- dspmqaut
- dmpmqaut
- rcrmqobj
- rcdmqimg
- dspmqfls
- amqtcert
- Managing keys and certificates

For an overview of channel security using SSL, see WebSphere MQ Security.

For details of MQSC commands associated with SSL, see the WebSphere MQ Script (MQSC) Command Reference.

For details of PCF commands associated with SSL, see WebSphere MQ Programmable Command Formats and Administration Interface.

Parent topic: Channel security

This build: January 26, 2011 11:54:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13310_

6.3.9. How authorizations work

The authorization specification tables in the topics in this section define precisely how the authorizations work and the restrictions that apply.

The tables apply to these situations:

- Applications that issue MQI calls
- Administration programs that issue MQSC commands as escape PCFs
- Administration programs that issue PCF commands

In this section, the information is presented as a set of tables that specify the following:

Action to be performed

MQI option, MQSC command, or PCF command.

Access control object

Queue, process, queue manager, namelist, authentication information, channel, client connection channel, listener, or service.

Authorization required

Expressed as an MQZAO_ constant.

In the tables, the constants prefixed by MQZAO_ correspond to the keywords in the authorization list for the setmqaut command for the particular entity. For example, MQZAO_BROWSE corresponds to the keyword +browse, MQZAO_SET_ALL_CONTEXT corresponds to the keyword +setall, and so on. These constants are defined in the header file cmqzc.h, supplied with the product.

Authorizations for MQI calls
MQCONN, MQOPEN, MQPUT1, and MQCLOSE might require authorization checks. The tables in this topic summarize the authorizations needed for each call.

Authorizations for MQSC commands in escape PCFs
This information summarizes the authorizations needed for each MQSC command contained in Escape PCF.

Authorizations for PCF commands
This section summarizes the authorizations needed for each PCF command.

Parent topic: Setting up security on UNIX systems and Windows

This build: January 26, 2011 11:54:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13320_

6.3.9.1. Authorizations for MQI calls

MQCONN, MQOPEN, MQPUT1, and MQCLOSE might require authorization checks. The tables in this topic summarize the authorizations needed for each call.

An application is allowed to issue specific MQI calls and options only if the user identifier under which it is running (or whose authorizations it is able to assume) has been granted the relevant authorization.

Four MQI calls might require authorization checks: MQCONN, MQOPEN, MQPUT1, and MQCLOSE.

For MQOPEN and MQPUT1, the authority check is made on the name of the object being opened, and not on the name, or names, resulting after a name has been resolved. For example, an application might be granted authority to open an alias queue without having authority to open the base queue to which the alias resolves. The rule is that the check is carried out on the first definition encountered during the process of resolving a name that is not a queue manager alias, unless the queue manager alias definition is opened directly; that is, its name is displayed in the *ObjectName* field of the object descriptor. Authority is always needed for the object being opened. In some cases additional queue-independent authority, obtained through an authorization for the queue manager object, is required.

Table 1, Table 2, Table 3, and Table 4 summarize the authorizations needed for each call. In the tables *Not applicable* means that authorization checking is not relevant to this operation; *No check* means that no authorization checking is performed.

Note: You will find no mention of namelists, channels, client connection channels, listeners, services, or authentication information objects in these tables. This is because none of the authorizations apply to these objects, except for MQOO_INQUIRE, for which the same authorizations apply as for the other objects.

The special authorization MQZAO_ALL_MQI includes all the authorizations in the tables that are relevant to the object type, except MQZAO_DELETE and MQZAO_DISPLAY, which are classed as administration authorizations.

Table 1. Security authorization needed for MQCONN calls

Authorization	Queue object (1)	Process object	Queue manager

required for:			object
MQCONN	Not applicable	Not applicable	MQZAO_CONNECT

Table 2. Security authorization needed for MQOPEN calls

Authorization required for:	Queue object (1)	Process object	Queue manager object
MQOO_INQUIRE	MQZAO_INQUIRE	MQZAO_INQUIRE	MQZAO_INQUIRE
MQOO_BROWSE	MQZAO_BROWSE	Not applicable	No check
MQOO_INPUT_*	MQZAO_INPUT	Not applicable	No check
MQOO_SAVE_ ALL_CONTEXT (2)	MQZAO_INPUT	Not applicable	Not applicable
MQOO_OUTPUT (Normal queue) (3)	MQZAO_OUTPUT	Not applicable	Not applicable
MQOO_PASS_ IDENTITY_CONTEXT (4)	MQZAO_PASS_ IDENTITY_CONTEXT	Not applicable	No check
MQOO_PASS_ALL_ CONTEXT (4, 5)	MQZAO_PASS _ALL_CONTEXT	Not applicable	No check
MQOO_SET_ IDENTITY_CONTEXT (4, 5)	MQZAO_SET_ IDENTITY_CONTEXT	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (6)
MQOO_SET_ ALL_CONTEXT (4, 7)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (6)
MQOO_OUTPUT (Transmission queue) (8)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (6)
MQOO_SET	MQZAO_SET	Not applicable	No check
MQOO_ALTERNATE_ USER_AUTHORITY	(9)	(9)	MQZAO_ALTERNATE_ USER_AUTHORITY (9, 10)

Table 3. Security authorization needed for MQPUT1 calls

Authorization required for:	Queue object (1)	Process object	Queue manager object
MQPMO_PASS_ IDENTITY_CONTEXT	MQZAO_PASS_ IDENTITY_CONTEXT (11)	Not applicable	No check
MQPMO_PASS_ALL _CONTEXT	MQZAO_PASS_ ALL_CONTEXT (11)	Not applicable	No check
MQPMO_SET_ IDENTITY_CONTEXT	MQZAO_SET_ IDENTITY_CONTEXT (11)	Not applicable	MQZAO_SET_ IDENTITY_CONTEXT (6)
MQPMO_SET_ ALL_CONTEXT	MQZAO_SET_ ALL_CONTEXT (11)	Not applicable	MQZAO_SET_ ALL_CONTEXT (6)
(Transmission queue) (8)	MQZAO_SET_ ALL_CONTEXT	Not applicable	MQZAO_SET_ ALL_CONTEXT (6)
MQPMO_ALTERNATE_ USER_AUTHORITY	(12)	Not applicable	MQZAO_ALTERNATE_ USER_AUTHORITY (10)

Table 4. Security authorization needed for MQCLOSE calls

Authorization required for:	Queue object (1)	Process object	Queue manager object
MQCO_DELETE	MQZAO_DELETE (13)	Not applicable	Not applicable
MQCO_DELETE _PURGE	MQZAO_DELETE (13)	Not applicable	Not applicable

Notes® for the tables:

1. **If opening a model queue:**
   - **MQZAO_DISPLAY authority is needed for the model queue, in addition to the authority to open the model queue for the type of access for which you are opening.**
   - **MQZAO_CREATE authority is not needed to create the dynamic queue.**
   - **The user identifier used to open the model queue is automatically granted all the queue-specific authorities (equivalent to MQZAO_ALL) for the dynamic queue created.**
2. **MQOO_INPUT_* must also be specified. This is valid for a local, model, or alias queue.**

3. This check is performed for all output cases, except transmission queues (see note 8).
4. MQOO_OUTPUT must also be specified.
5. MQOO_PASS_IDENTITY_CONTEXT is also implied by this option.
6. This authority is required for both the queue manager object and the particular queue.
7. MQOO_PASS_IDENTITY_CONTEXT, MQOO_PASS_ALL_CONTEXT, and MQOO_SET_IDENTITY_CONTEXT are also implied by this option.
8. This check is performed for a local or model queue that has a *Usage* queue attribute of MQUS_TRANSMISSION, and is being opened directly for output. It does not apply if a remote queue is being opened (either by specifying the names of the remote queue manager and remote queue, or by specifying the name of a local definition of the remote queue).
9. At least one of MQOO_INQUIRE (for any object type), or MQOO_BROWSE, MQOO_INPUT_*, MQOO_OUTPUT, or MQOO_SET (for queues) must also be specified. The check carried out is as for the other options specified, using the supplied alternate-user identifier for the specific-named object authority, and the current application authority for the MQZAO_ALTERNATE_USER_IDENTIFIER check.
10. This authorization allows any *AlternateUserId* to be specified.
11. An MQZAO_OUTPUT check is also carried out if the queue does not have a *Usage* queue attribute of MQUS_TRANSMISSION.
12. The check carried out is as for the other options specified, using the supplied alternate-user identifier for the specific-named queue authority, and the current application authority for the MQZAO_ALTERNATE_USER_IDENTIFIER check.
13. The check is carried out only if both of the following are true:
    - A permanent dynamic queue is being closed and deleted.
    - The queue was not created by the MQOPEN call that returned the object handle being used.

    Otherwise, there is no check.

Parent topic: »How authorizations work«

This build: January 26, 2011 11:54:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13330_

6.3.9.2. Authorizations for MQSC commands in escape PCFs

This information summarizes the authorizations needed for each MQSC command contained in Escape PCF.

Not applicable means that authorization checking is not relevant to this operation.

The user ID under which the program that submits the command is running must also have the following authorities:

- MQZAO_CONNECT authority to the queue manager
- DISPLAY authority on the queue manager in order to perform PCF commands
- Authority to issue the MQSC command within the text of the Escape PCF command

ALTER *object*

Object	Authorization required
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	MQZAO_CHANGE
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

CLEAR *object*

Object	Authorization required
Queue	MQZAO_CLEAR
Topic	MQZOA_CLEAR
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

DEFINE *object* NOREPLACE (1)

Object	Authorization required
Queue	MQZAO_CREATE (2)
Topic	MQZAO_CREATE (2)
Process	MQZAO_CREATE (2)
Queue manager	Not applicable
Namelist	MQZAO_CREATE (2)
Authentication information	MQZAO_CREATE (2)
Channel	MQZAO_CREATE (2)
Client connection channel	MQZAO_CREATE (2)
Listener	MQZAO_CREATE (2)
Service	MQZAO_CREATE (2)

DEFINE *object* REPLACE (1, 3)

Object	Authorization required
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

DELETE *object*

Object	Authorization required
Queue	MQZAO_DELETE
Topic	MQZAO_DELETE
Process	MQZAO_DELETE
Queue manager	Not applicable
Namelist	MQZAO_DELETE
Authentication information	MQZAO_DELETE
Channel	MQZAO_DELETE
Client connection channel	MQZAO_DELETE
Listener	MQZAO_DELETE
Service	MQZAO_DELETE

DISPLAY *object*

Object	Authorization required
Queue	MQZAO_DISPLAY
Topic	MQZAO_DISPLAY

Process	MQZAO_DISPLAY
Queue manager	MQZAO_DISPLAY
Namelist	MQZAO_DISPLAY
Authentication information	MQZAO_DISPLAY
Channel	MQZAO_DISPLAY
Client connection channel	MQZAO_DISPLAY
Listener	MQZAO_DISPLAY
Service	MQZAO_DISPLAY

PING CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

RESET CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

RESOLVE CHANNEL

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

START CHANNEL/LISTENER/SERVICE

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable

Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	MQZAO_CONTROL

STOP CHANNEL/LISTENER/SERVICE

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	MQZAO_CONTROL

Note:

1. **For DEFINE commands, MQZAO_DISPLAY authority is also needed for the LIKE object if one is specified, or on the appropriate SYSTEM.DEFAULT.xxx object if LIKE is omitted.**
2. **The MQZAO_CREATE authority is not specific to a particular object or object type. Create authority is granted for all objects for a specified queue manager, by specifying an object type of QMGR on the setmqaut command.**
3. **This applies if the object to be replaced already exists. If it does not, the check is as for DEFINE *object* NOREPLACE.**

Parent topic: »How authorizations work«

This build: January 26, 2011 11:54:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13340_

6.3.9.3. Authorizations for PCF commands

This section summarizes the authorizations needed for each PCF command.

***No check* means that no authorization checking is carried out; *Not applicable* means that authorization checking is not relevant to this operation.**

The user ID under which the program that submits the command is running must also have the following authorities:

- **MQZAO_CONNECT authority to the queue manager**
- **DISPLAY authority on the queue manager in order to perform PCF commands**

The special authorization MQZAO_ALL_ADMIN includes all the authorizations in the following list that are relevant to the object type, except MQZAO_CREATE, which is not specific to a particular object or object type.

Change *object*

Object	Authorization required
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	MQZAO_CHANGE
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE

Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Clear *object*

Object	Authorization required
Queue	MQZAO_CLEAR
	MQZAO_CLEAR
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Copy *object* (without replace) (1)

Object	Authorization required
Queue	MQZAO_CREATE (2)
Topic	MQZAO_CREATE (2)
Process	MQZAO_CREATE (2)
Queue manager	Not applicable
NamelistMQZAO_CREATE	MQZAO_CREATE (2)
Authentication information	MQZAO_CREATE (2)
Channel	MQZAO_CREATE (2)
Client connection channel	MQZAO_CREATE (2)
Listener	MQZAO_CREATE (2)
Service	MQZAO_CREATE (2)

Copy *object* (with replace) (1, 4)

Object	Authorization required
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Create *object* (without replace) (3)

Object	Authorization required
Queue	MQZAO_CREATE (2)
Topic	MQZAO_CREATE (2)
Process	MQZAO_CREATE (2)
Queue manager	Not applicable
Namelist	MQZAO_CREATE (2)
Authentication information	MQZAO_CREATE (2)
Channel	MQZAO_CREATE (2)
Client connection channel	MQZAO_CREATE (2)
Listener	MQZAO_CREATE (2)
Service	MQZAO_CREATE (2)

Create *object* (with replace) (3, 4)

Object	Authorization required
Queue	MQZAO_CHANGE
Topic	MQZAO_CHANGE
Process	MQZAO_CHANGE
Queue manager	Not applicable
Namelist	MQZAO_CHANGE
Authentication information	MQZAO_CHANGE
Channel	MQZAO_CHANGE
Client connection channel	MQZAO_CHANGE
Listener	MQZAO_CHANGE
Service	MQZAO_CHANGE

Delete *object*

Object	Authorization required
Queue	MQZAO_DELETE
Topic	MQZAO_DELETE
Process	MQZAO_DELETE
Queue manager	MQZAO_DELETE
Namelist	MQZAO_DELETE
Authentication information	MQZAO_DELETE
Channel	MQZAO_DELETE
Client connection channel	MQZAO_DELETE
Listener	MQZAO_DELETE
Service	MQZAO_DELETE

Inquire *object*

Object	Authorization required
Queue	MQZAO_DISPLAY
Topic	MQZAO_DISPLAY
Process	MQZAO_DISPLAY
Queue manager	MQZAO_DISPLAY
Namelist	MQZAO_DISPLAY
Authentication information	MQZAO_DISPLAY
Channel	MQZAO_DISPLAY
Client connection channel	MQZAO_DISPLAY
Listener	MQZAO_DISPLAY
Service	MQZAO_DISPLAY

Inquire *object* names

Object	Authorization required
Queue	No check
Topic	No check
Process	No check
Queue manager	No check
Namelist	No check
Authentication information	No check
Channel	No check
Client connection channel	No check
Listener	No check
Service	No check

Ping Channel

Object	Authorization required
Queue	Not applicable

Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Reset Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Reset Queue Statistics

Object	Authorization required
Queue	MQZAO_DISPLAY and MQZAO_CHANGE
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	Not applicable
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Resolve Channel

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL_EXTENDED
Client connection channel	Not applicable
Listener	Not applicable
Service	Not applicable

Start Channel/Listener/Service

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable

Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	MQZAO_CONTROL

Stop Channel/Listener/Service

Object	Authorization required
Queue	Not applicable
Topic	Not applicable
Process	Not applicable
Queue manager	Not applicable
Namelist	Not applicable
Authentication information	Not applicable
Channel	MQZAO_CONTROL
Client connection channel	Not applicable
Listener	MQZAO_CONTROL
Service	MQZAO_CONTROL

Note:

1. **For Copy commands, MQZAO_DISPLAY authority is also needed for the From object.**
2. **The MQZAO_CREATE authority is not specific to a particular object or object type. Create authority is granted for all objects for a specified queue manager, by specifying an object type of QMGR on the setmqaut command.**
3. **For Create commands, MQZAO_DISPLAY authority is also needed for the appropriate SYSTEM.DEFAULT.* object.**
4. **This applies if the object to be replaced already exists. If it does not, the check is as for Copy or Create without replace.**

Parent topic: »How authorizations work«

This build: January 26, 2011 11:55:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13350_

6.3.10. Special considerations for security on Windows

»Some security functions behave differently on different version of Windows.«

WebSphere® MQ for Windows runs on Windows 2003, Windows Vista, Windows Server 2008, and Windows XP, but the operation of WebSphere MQ security can be affected by differences between the platforms.

WebSphere MQ security relies on calls to the operating system API for information about user authorizations and group memberships. Some functions do not behave identically on the Windows systems. This collection of topics includes descriptions of how those differences might affect WebSphere MQ security when you are running WebSphere MQ in a Windows environment.

»When you get a 'group not found' error on Windows«
This problem can arise because WebSphere MQ loses access to the local mqm group when Windows servers are promoted to, or demoted from, domain controllers. To remedy this problem, re-create the local mqm group.

»When you have problems with WebSphere MQ and domain controllers on Windows«
Certain problems can arise with security settings when Windows servers are promoted to domain controllers.

»Applying security template files to Windows«
Applying a template might affect the security settings applied to WebSphere MQ files and directories. If you use the highly secure template, apply it before installing WebSphere

MQ.

»Nested groups«
There are restrictions on the use of nested groups. These result partly from the domain functional level and partly from WebSphere MQ restrictions.

Parent topic: »Setting up security on UNIX systems and Windows«

This build: January 26, 2011 11:55:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13360_

6.3.10.1. When you get a 'group not found' error on Windows

»This problem can arise because WebSphere® MQ loses access to the local mqm group when Windows servers are promoted to, or demoted from, domain controllers. To remedy this problem, re-create the local mqm group. «

The symptom is an error indicating the lack of a local mqm group, for example:

```
>crtmqm qm0
AMQ8066:Local mqm group not found.
```

Altering the state of a machine between server and domain controller can affect the operation of WebSphere MQ, because WebSphere MQ uses a locally-defined mqm group. When a server is promoted to be a domain controller, the scope changes from local to domain local. When the machine is demoted to server, all domain local groups are removed. This means that changing a machine from server to domain controller and back to server loses access to a local mqm group.

To remedy this problem, re-create the local mqm group using the standard Windows management tools. Because all group membership information is lost, you must reinstate privileged WebSphere MQ users in the newly-created local mqm group. If the machine is a domain member, you must also add the domain mqm group to the local mqm group to grant privileged domain WebSphere MQ user IDs the required level of authority.

Parent topic: »Special considerations for security on Windows«

This build: January 26, 2011 11:55:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13370_

6.3.10.2. When you have problems with WebSphere MQ and domain controllers on Windows

Certain problems can arise with security settings when Windows servers are promoted to domain controllers.

While promoting Windows 2000, Windows 2003, or Windows Server 2008 servers to domain controllers, you are presented with the option of selecting a default or non-default security setting relating to user and group permissions. This option controls whether arbitrary users are able to retrieve group memberships from the active directory. Because WebSphere® MQ relies on group membership information to implement its security policy, it is important that the user ID that is performing WebSphere MQ operations can determine the group memberships of other users.

On Windows 2000, when a domain is created using the default security option, the default user ID created by WebSphere MQ during the installation process (MUSR_MQADMIN) can obtain group memberships for other users as required. The product then installs normally, creating default objects, and the queue manager can determine the access authority of local and domain users if required.

On Windows 2000, when a domain is created using the non-default security option, or on Windows 2003 and Windows Server 2008 when a domain is created using the default security option, the user ID created by WebSphere MQ during the installation (MUSR_MQADMIN) cannot always determine the required group memberships. In this case, you need to know:

- How Windows 2000 with non-default, or Windows 2003 and Windows Server 2008 with default,

security permissions behaves

- How to allow domain mqm group members to read group membership
- How to configure WebSphere MQ Services to run under a domain user

»Windows 2000 domain with non-default, or Windows 2003 and Windows Server 2008 domain with default, security permissions«
Installation of WebSphere MQ behaves differently on these operating systems depending on whether a local user or domain user performs the installation.

»Configuring WebSphere MQ Services to run under a domain user on Windows«
Use the Prepare WebSphere MQ wizard to enter the account details of the domain user account. Alternatively, you can use the AMQMSRVN command.

Parent topic: »Special considerations for security on Windows«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13380_

6.3.10.2.1. Windows 2000 domain with non-default, or Windows 2003 and Windows Server 2008 domain with default, security permissions

Installation of WebSphere MQ behaves differently on these operating systems depending on whether a local user or domain user performs the installation.

If a local user installs WebSphere® MQ, the Prepare WebSphere MQ Wizard detects that the local user (MUSR_MQADMIN) created for the WebSphere MQ services (AMQMSRVN) can retrieve the group membership information of the installing user. The Prepare WebSphere MQ Wizard asks the user questions about the network configuration to determine whether there are other user accounts defined on domain controllers running on Windows 2000 or later. If so, the WebSphere MQ services need to run under a domain user account with particular settings and authorities. The Prepare WebSphere MQ Wizard prompts the user for the account details of this user. Its online help provides details of the domain user account required that can be sent to the domain administrator.

If a domain user installs WebSphere MQ, the Prepare WebSphere MQ Wizard detects that the local user (MUSR_MQADMIN) created for the WebSphere MQ services (AMQMSRVN) cannot retrieve the group membership information of the installing user. In this case, the Prepare WebSphere MQ Wizard always prompts the user for the account details of the domain user account for the WebSphere MQ services to use.

When WebSphere MQ services needs to use a domain user account, WebSphere MQ cannot operate correctly until this has been configured using the Prepare WebSphere MQ Wizard. This configuration includes creating default objects such as the Default Configuration. The Prepare WebSphere MQ Wizard does not allow the user to continue with other tasks, such as creating the Default Configuration, until the WebSphere MQ services have been configured with a suitable account.

If a Windows 2000 domain has been configured with non-default security permissions, the usual solution to enable WebSphere MQ to work correctly is to configure it with a suitable domain user account, as described above.

See »Creating and setting up domain accounts for WebSphere MQ« for more information.

Parent topic: »When you have problems with WebSphere MQ and domain controllers on Windows«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13390_

6.3.10.2.2. Configuring WebSphere MQ Services to run under a domain user on Windows

Use the Prepare WebSphere® MQ wizard to enter the account details of the domain user account. Alternatively, you can use the AMQMSRVN command.

If you use AMQMSRVN, enter the following command line to set the domain user account:

```
AMQMSRVN -user [domain]\[userid] -password [password]
```

In either case, WebSphere MQ allocates the correct security rights and group membership to the new user account.

Parent topic: »When you have problems with WebSphere MQ and domain controllers on Windows«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13400_

6.3.10.3. Applying security template files to Windows

»Applying a template might affect the security settings applied to WebSphere® MQ files and directories. If you use the highly secure template, apply it before installing WebSphere MQ.«

Windows supports text-based security template files that you can use to apply uniform security settings to one or more computers with the Security Configuration and Analysis MMC snap-in. In particular, Windows supplies several templates that include a range of security settings with the aim of providing specific levels of security. These templates include Compatible, Secure, and Highly Secure.

Applying one of these templates might affect the security settings applied to WebSphere MQ files and directories. If you want to use the Highly Secure template, configure your machine before you install WebSphere MQ.

If you apply the highly secure template to a machine on which WebSphere MQ is already installed, all the permissions you have set on the WebSphere MQ files and directories are removed. Because these permissions are removed, you lose *Administrator*, *mqm*, and, when applicable, *Everyone* group access from the error directories.

Parent topic: »Special considerations for security on Windows«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13410_

6.3.10.4. Nested groups

There are restrictions on the use of nested groups. These result partly from the domain functional level and partly from WebSphere MQ restrictions.

Active Directory can support different group types within a Domain context depending on the Domain functional level. By default, Windows 2003 domains are in the *Windows 2000 mixed* functional level. (Windows server 2003 , Windows XP, Windows Vista, and Windows Server 2008 all follow the Windows 2003 domain model.) The domain functional level determines the supported group types and level of nesting allowed when configuring user IDs in a domain environment. Refer to Active Directory documentation for details on the Group Scope and inclusion criteria.

In addition to Active Directory requirements, further restrictions are imposed on IDs used by WebSphere MQ. The network APIs used by WebSphere MQ do not support all the configurations that are supported by the domain functional level. As a result, WebSphere MQ is not able to query the group memberships of any Domain IDs present in a Domain Local group which is then nested in a local group. Furthermore, multiple nesting of global and universal groups is not supported. However, immediately nested global or universal groups are supported.

Parent topic: »Special considerations for security on Windows«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13420_

6.4. Transactional support

Enable your applications to work reliably with databases.

This chapter introduces transactional support. The work required to enable your applications to use WebSphere® MQ in conjunction with a database product spans the areas of application programming and system administration. Use the information here together with Chapter 13, "Committing and backing out units of work" of the WebSphere MQ Application Programming Guide.

We start by introducing the units of work that form transactions, then describe the ways in which you enable WebSphere MQ to coordinate transactions with databases.

Introducing units of work
This topic introduces and defines the general concepts of unit of work, commit, backout and sync point. It also contains two scenarios illustrating global units of work.

Scenario 1: Queue manager performs the coordination

Scenario 2: Other software provides the coordination

Unit of recovery disposition
WebSphere MQ for z/OS provides unit of recovery dispositions. This feature allows you to configure whether the second phase of 2-phase commit transactions can be driven, for example, during recovery, when connected to another queue manager within the same queue-sharing group (QSG).

Parent topic: Configuring and managing WebSphere MQ

This build: January 26, 2011 11:55:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13430_

6.4.1. Introducing units of work

This topic introduces and defines the general concepts of unit of work, commit, backout and sync point. It also contains two scenarios illustrating global units of work.

When a program puts messages on queues within a unit of work, those messages are made visible to other programs only when the program *commits* the unit of work. To commit a unit of work, all updates must be successful to preserve data integrity.

If the program detects an error and decides not to make the put operation permanent, it can *back out* the unit of work. When a program performs a backout, WebSphere® MQ restores the queues by removing the messages that were put on the queues by that unit of work.

Similarly, when a program gets messages from one or more queues within a unit of work, those messages remain on the queues until the program commits the unit of work, but the messages are not available to be retrieved by other programs. The messages are permanently deleted from the queues when the program commits the unit of work. If the program backs out the unit of work, WebSphere MQ restores the queues by making the messages available to be retrieved by other programs.

The decision to commit or back out the changes is taken, in the simplest case, at the end of a task. However, it can be more useful for an application to synchronize data changes at other logical points within a task. These logical points are called sync points (or synchronization points) and the period of processing a set of updates between two sync points is called a *unit of work*. Several MQGET calls and MQPUT calls can be part of a single unit of work.

With WebSphere MQ, we need to distinguish between *local* and *global* units of work:

Local units of work

Are those in which the only actions are puts to, and gets from, WebSphere MQ queues, and the coordination of each unit of work is provided within the queue manager using a *single-phase commit* process.

Use local units of work when the only resources to be updated are the queues managed by a single WebSphere MQ queue manager. Updates are committed using the MQCMIT verb or backed out using MQBACK.

There are no system administration tasks, other than log management, involved in using local units of work. In your applications, where you use the MQPUT and MQGET calls with MQCMIT and MQBACK, try using the MQPMO_SYNCPOINT and MQGMO_SYNCPOINT options. (For information on log management, see Managing log files.)

Global units of work

Are those in which other resources, such as tables in a relational database, are also updated. When more than one *resource manager* is involved, there is a need for *transaction manager* software that uses a *two-phase commit* process to coordinate the global unit of work.

Use global units of work when you also need to include updates to relational database manager software, such as DB2®, Oracle, Sybase, and Informix®.

We define two scenarios for global units of work:

1. In the first, the queue manager itself acts as the transaction manager. In this scenario, MQI verbs control the global units of work; they are started in applications using the MQBEGIN verb and then committed using MQCMIT or backed out using MQBACK.
2. In the second, the transaction manager role is performed by other software, such as TXSeries, Encina, or Tuxedo. In this scenario, an API provided by the transaction manager software is used to control the unit of work (for example, EXEC CICS® SYNCPOINT for TXSeries).

The following sections describe all the steps necessary to use global units of work, organized by the two scenarios:

- Scenario 1: Queue manager performs the coordination
- Scenario 2: Other software provides the coordination

Parent topic: »Transactional support«

This build: January 26, 2011 11:55:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13440_

6.4.2. Scenario 1: Queue manager performs the coordination

This section describes this scenario, including:

- Database coordination
- DB2 configuration
- Oracle configuration
- Informix configuration
- Sybase configuration
- Multiple database configurations
- Security considerations
- Administration tasks

»Database coordination«

»DB2 configuration«

»Oracle configuration«

»Informix configuration«

»Sybase configuration«

»Multiple database configurations«

»Security considerations«

»Administration tasks«

»XA dynamic registration«

Parent topic: »Transactional support«

This build: January 26, 2011 11:55:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13450_

6.4.2.1. Database coordination

When the queue manager coordinates global units of work itself, it becomes possible to integrate database updates within the units of work. That is, a mixed MQI and SQL application can be written, and the MQCMIT and MQBACK verbs can be used to commit or roll back the changes to the queues and databases together.

The queue manager achieves this using the two-phase commit protocol described in *X/Open Distributed Transaction Processing: The XA Specification*. When a unit of work is to be committed, the queue manager first asks each participating database manager whether it is prepared to commit its updates. Only if all the participants, including the queue manager itself, are prepared to commit, are all the queue and database updates committed. If any participant cannot prepare its updates, the unit of work is backed out instead.

In general, a global unit of work is implemented in an application by the following method (in pseudocode):

```
MQBEGIN
MQGET (include the flag MQGMO_SYNCPOINT in the message options)
MQPUT (include the flag MQPMO_SYNCPOINT in the message options)
SQL INSERT
MQCMIT
```

The purpose of MQBEGIN is to denote the beginning of a global unit of work. The purpose of MQCMIT is to denote the end of the global unit of work, and to complete it with all participating resource managers, using the two-phase commit protocol.

When the unit of work (also known as a *transaction*) is completed successfully using MQCMIT, all actions taken within that unit of work are made permanent or irreversible. If, for any reason, the unit of work fails, all actions are instead backed out. It is not acceptable for one action comprising a unit of work to be made permanent while another is forgotten. This is the principle of a unit of work: either all actions within the unit of work are made permanent or none of them are.

Note:

1. The application programmer can force a unit of work to be backed out by calling MQBACK. The unit of work is also backed out by the queue manager if the application or database *crashes* before MQCMIT is called.
2. If an application calls MQDISC without calling MQCMIT, the queue manager behaves as if MQCMIT had been called, and commits the unit of work.

In between MQBEGIN and MQCMIT, the queue manager does not make any calls to the database to update its resources. That is, the only way a database's tables are changed is by your code (for example, the SQL INSERT in the pseudocode above).

Full recovery support is provided if the queue manager loses contact with any of the database managers during the commit protocol. If a database manager becomes unavailable while it is in doubt, that is, it has successfully prepared to commit, but has yet to receive a commit or backout decision, the queue manager remembers the outcome of the unit of work until that outcome has been successfully delivered to the database. Similarly, if the queue manager terminates with incomplete commit operations outstanding, these are remembered over queue manager restart. If an application terminates unexpectedly, the integrity of the unit of work is not compromised, but the outcome depends on where in the process the application terminated, as described inTable 2.

What happens when the database or application program crashes is summarized in the tables below:

Table 1. What happens when a database server crashes

Before the application call to MQCMIT.	The unit of work is backed out.
During the application call to MQCMIT,	The unit of work is backed out with a

before all databases have indicated that they have successfully prepared.	reason code of MQRC_BACKED_OUT.
During the application call to MQCMIT, **after** all databases have indicated that they have successfully prepared, but before all have indicated that they have successfully committed.	The unit of work is held in recoverable state by the queue manager, with a reason code of MQRC_OUTCOME_PENDING.
During the application call to MQCMIT, **after** all databases have indicated that they have successfully committed.	The unit of work is committed with a reason code of MQRC_NONE.
After the application call to MQCMIT.	The unit of work is committed with a reason code of MQRC_NONE.

Table 2. What happens when an application program crashes

Before the application call to MQCMIT.	The unit of work is backed out.
During the application call to MQCMIT, **before** the queue manager has received the application's MQCMIT request.	The unit of work is backed out.
During the application call to MQCMIT, **after** the queue manager has received the application's MQCMIT request.	The queue manager tries to commit using two-phase commit (subject to the database products successfully executing and committing their parts of the unit of work).

In the case where the reason code on return from MQCMIT is MQRC_OUTCOME_PENDING, the unit of work is remembered by the queue manager until it has been able to reestablish contact with the database server, and tell it to commit its part of the unit of work. Refer to Administration tasks for information on how and when recovery is done.

The queue manager communicates with database managers using the XA interface as described in *X/Open Distributed Transaction Processing: The XA Specification*. Examples of these function calls are xa_open, xa_start, xa_end, xa_prepare, and xa_commit. We use the terms *transaction manager* and *resource manager* in the same sense as they are used in the XA specification.

»Restrictions«

»Switch load files«

»Configuring your system for database coordination«

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13460_

6.4.2.1.1. Restrictions

The following restrictions apply to the database coordination support:

- **The ability to coordinate database updates within WebSphere® MQ units of work is not supported in an MQI client application. The use of MQBEGIN in a client application fails, as described in the WebSphere MQ Application Programming Reference. A program that calls MQBEGIN must run as a *server* application on the same machine as the queue manager. Note: A *server* application is a program that has been linked with the necessary WebSphere MQ server libraries; a *client* application is a program that has been linked with the necessary WebSphere MQ client libraries. See WebSphere MQ Clients and the WebSphere MQ Application Programming Guide for details on compiling and linking your programs.**
- **The database server can reside on a different machine from the queue manager server, as long as the database client is installed on the same machine as the queue manager, and it supports this function. Consult the database product's documentation to determine whether their client software can be used for two-phase commit systems.**
- **Although the queue manager behaves as a resource manager (for the purposes of being involved in Scenario 2 global units of work), it is not possible to make one queue manager coordinate another queue manager within its Scenario 1 global units of work.**

Parent topic: »Database coordination«

This build: January 26, 2011 11:55:02

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13470_

6.4.2.1.2. Switch load files

The switch load file is a shared library (a DLL on Windows systems) that is loaded by the code in your WebSphere® MQ application and the queue manager. Its purpose is to simplify the loading of the database's client shared library, and to return the pointers to the XA functions.

The details of the switch load file must be specified before the queue manager is started. The details are placed in the qm.ini file (UNIX systems), or the Registry (Windows systems).

- On Windows or Linux (x86 platform) systems use the WebSphere MQ Explorer. On Windows systems the Registry is updated. On Linux (x86 platform) systems the file, qm.ini, is updated.
- On all other systems edit the file, qm.ini, directly.

The C source for the switch load file is supplied with the WebSphere MQ installation if it supports Scenario 1 global units of work. The source contains a function called MQStart. When the switch load file is loaded, the queue manager calls this function, which returns the address of a structure called an *XA switch*.

The XA switch structure exists in the database client shared library, and contains a number of function pointers, as described in Table 1:

Table 1. XA switch function pointers

Function pointer name	XA function	Purpose
xa_open_entry	xa_open	Connect to database
xa_close_entry	xa_close	Disconnect from database
xa_start_entry	xa_start	Start a branch of a global unit of work
xa_end_entry	xa_end	Suspend a branch of a global unit of work
xa_rollback_entry	xa_rollback	Roll back a branch of a global unit of work
xa_prepare_entry	xa_prepare	Prepare to commit a branch of a global unit of work
xa_commit_entry	xa_commit	Commit a branch of a global unit of work
xa_recover_entry	xa_recover	Discover from the database whether it has an in-doubt unit of work
xa_forget_entry	xa_forget	Allow a database to forget a branch of a global unit of work
xa_complete_entry	xa_complete	Complete a branch of a global unit of work

During the first MQBEGIN call in your application, the WebSphere MQ code that executes as part of MQBEGIN loads the switch load file, and calls the xa_open function in the database shared library. Similarly, during queue manager startup, and on other subsequent occasions, some queue manager processes load the switch load file and call xa_open.

You can reduce the number of xa_* calls by using *dynamic registration*. For a complete description of this optimization technique, see XA dynamic registration.

Parent topic: »Database coordination«

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13480_

6.4.2.1.3. Configuring your system for database coordination

There are several tasks that you must perform before a database manager can participate in global units of works coordinated by the queue manager. These are described here as follows:

- Installing and configuring the database product

- Creating switch load files
- Adding configuration information to the queue manager
- Writing and modifying your applications
- Testing the system

Installing and configuring the database product

Creating switch load files

Adding configuration information to the queue manager

Writing and modifying your applications

Testing the system

Parent topic: Database coordination

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13490_

6.4.2.1.3.1. Installing and configuring the database product

To install and configure your database product, see the product's own documentation. This topics in this section describe general configuration issues and how they relate to interoperation between WebSphere® MQ and the database.

Database connections

Client/server configuration

Parent topic: Configuring your system for database coordination

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13500_

6.4.2.1.3.1.1. Database connections

An application that establishes a standard connection to the queue manager is associated with a thread in a separate local queue manager agent process. (A connection that is not a *fastpath* connection is a *standard* connection in this context. For more information, see "Connecting to a queue manager using the MQCONNX call" in the WebSphere MQ Application Programming Guide.)

When the application issues MQBEGIN, both it and the agent process call the xa_open function in the database client library. In response to this, the database client library code *connects* to the database that is to be involved in the unit of work *from both the application and queue manager processes*. These database connections are maintained as long as the application remains connected to the queue manager.

This is an important consideration if the database supports only a limited number of users or connections, because two connections are being made to the database to support the one application program.

Parent topic: Installing and configuring the database product

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa13510_

6.4.2.1.3.1.2. Client/server configuration

The database client library that is loaded into the WebSphere® MQ queue manager and application processes must be able to send to and receive from its server. Ensure that:

- The database's client/server configuration files have the correct details
- The relevant environment variables are set in the environment of the queue manager and the application processes

Parent topic: »Installing and configuring the database product«

This build: January 26, 2011 11:55:02

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13520_

6.4.2.1.3.2. Creating switch load files

WebSphere® MQ comes with a sample makefile, used to build switch load files for the supported database managers. This makefile, together with all the associated C source files required to build the switch load files, is installed in the following directories:

- For WebSphere MQ for Windows, in the C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm\ directory
- For WebSphere MQ for UNIX systems, in the /opt/mqm/samp/xatm/ directory (/usr/mqm/samp/xatm on AIX®)

The sample source modules used to build the switch load files are:

- For DB2®, db2swit.c
- For Oracle, oraswit.c
- For Informix®, infswit.c
- For Sybase, sybswit.c

When you generate switch load files it is recommended that 32-bit switch load files are installed in /var/mqm/exits and 64-bit switch load files are installed in /var/mqm/exits64.

If you have 32-bit queue managers then the sample make file, xaswit.mak, will install a 32-bit switch load file in /var/mqm/exits.

If you have 64-bit queue managers then the sample make file, xaswit.mak, will install a 32-bit switch load file in /var/mqm/exits, and a 64-bit switch load file in /var/mqm/exits64.

Parent topic: »Configuring your system for database coordination«

This build: January 26, 2011 11:55:02

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13530_

6.4.2.1.3.3. Adding configuration information to the queue manager

When you have created a switch load file for your database manager, and placed it in a safe location, you must specify that location to your queue manager.

- On Windows and Linux (x86 platform) systems use the WebSphere® MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Add an XAResourceManager stanza for the database that your queue manager is going to coordinate. The most common case is for there to be only one database, and therefore only one XAResourceManager stanza. More complicated configurations involving multiple databases, are discussed in Multiple database configurations. The attributes of the XAResourceManager stanza are as follows:

Name=name

User-chosen string that identifies the resource manager. In effect, it gives a name to the XAResourceManager stanza. The name is mandatory and can be up to 31 characters in length.

The name you choose must be unique; there must be only one XAResourceManager stanza with this name in this qm.ini file. The name should also be meaningful, because the queue manager uses it to refer to this resource manager both in queue manager error log messages and in output when the dspmqtrn command is used. (See Displaying outstanding units of work with the dspmqtrn command for more information.)

Once you have chosen a name, and have started the queue manager, do not change the Name attribute. This is discussed in more detail in Changing configuration information.

SwitchFile=name

This is the name of the XA switch load file you built earlier. This is a mandatory attribute. The code in the queue manager and WebSphere MQ application processes tries to load the switch load file on two occasions:

1. At queue manager startup
2. When you make the first call to MQBEGIN in your WebSphere MQ application process

The security and permissions attributes of your switch load file must allow these processes to perform this action.

XAOpenString=string

This is a string of data that WebSphere MQ code passes in its calls to the database manager's xa_open function. This is an optional attribute; if it is omitted a zero-length string is assumed.

The code in the queue manager and WebSphere MQ application processes call the xa_open function on two occasions:

1. At queue manager startup
2. When you make the first call to MQBEGIN in your WebSphere MQ application process

The format for this string is particular to each database product, and will be described in the documentation for that product. In general, the xa_open string contains authentication information (user name and password) to allow a connection to the database in both the queue manager and the application processes.

XACloseString=string

This is a string of data that WebSphere MQ code passes in its calls to the database manager's xa_close function. This is an optional attribute; if it is omitted a zero-length string is assumed.

The code in the queue manager and WebSphere MQ application processes call the xa_close function on two occasions:

1. At queue manager startup
2. When you make a call to MQDISC in your WebSphere MQ application process, having earlier made a call to MQBEGIN

The format for this string is particular to each database product, and will be described in the documentation for that product. In general, the string is empty, and it is common to omit the XACloseString attribute from the XAResourceManager stanza.

ThreadOfControl=THREAD|PROCESS

The ThreadOfControl value can be THREAD or PROCESS. The queue manager uses it for serialization purposes. This is an optional attribute; if it is omitted, the value PROCESS is assumed.

If the database client code allows threads to call the XA functions without serialization, the value for ThreadOfControl can be THREAD. The queue manager assumes that it can call the XA functions in the database client shared library from multiple threads at the same time, if necessary.

If the database client code does not allow threads to call its XA functions in this way, the value for ThreadOfControl must be PROCESS. In this case, the queue manager serializes all calls to the database client shared library so that only one call at a time is made from within a particular process. You probably also need to ensure that your application performs similar serialization if it runs with multiple threads.

Note that this issue, of the database product's ability to cope with multi-threaded processes in this way, is an issue for that product's vendor. Consult the database product's documentation for details on whether you can set the ThreadOfControl attribute to THREAD or PROCESS. We recommend that, if you can, you set ThreadOfControl to THREAD. If in doubt, the *safer* option is to set it to PROCESS, although you will lose the potential performance benefits of using THREAD.

Parent topic: »Configuring your system for database coordination«

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13540_

6.4.2.1.3.4. Writing and modifying your applications

The sample application programs for Scenario 1 global units of work that are supplied with a WebSphere® MQ installation are described in the WebSphere MQ Application Programming Guide.

In general, a global unit of work is implemented in an application by the following method (in pseudocode):

```
MQBEGIN
MQGET
MQPUT
SQL INSERT
MQCMIT
```

The purpose of MQBEGIN is to denote the beginning of a global unit of work. The purpose of MQCMIT is to denote the end of the global unit of work, and to complete it with all participating resource managers, using the two-phase commit protocol.

In between MQBEGIN and MQCMIT, the queue manager does not make any calls to the database to update its resources. That is, the only way a database's tables are changed is by your code (for example, the SQL INSERT in the pseudocode above).

The role of the queue manager, as far as the database is concerned, is to tell it when a global unit of work has started, when it has ended, and whether the global unit of work should be committed or rolled-back.

As far as your application is concerned, the queue manager performs two roles: a resource manager (where the resources are messages on queues) and the transaction manager for the global unit of work.

We recommend that you start with the supplied sample programs, and work through the various WebSphere MQ and database API calls that are being made in those programs. The API calls concerned are fully documented in the WebSphere MQ Application Programming Guide, the WebSphere MQ Application Programming Reference, and (in the case of the database's own API) the database's own documentation.

Parent topic: »Configuring your system for database coordination«

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13550_

6.4.2.1.3.5. Testing the system

You only know whether your application and system are correctly configured by running them during testing. You can test the system's configuration (the successful communication between queue manager and database) by building and running one of the supplied sample programs.

Parent topic: »Configuring your system for database coordination«

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13560_

6.4.2.2. DB2 configuration

The supported levels of DB2® are defined on the WebSphere MQ detailed system requirements page.

Note: 32-bit instances of DB2 are not supported on platforms where the queue manager is 64-

bit.

Do the following:

1. Check the environment variable settings.
2. Create the DB2 switch load file.
3. Add resource manager configuration information.
4. Change DB2 configuration parameters if necessary.

Read this information in conjunction with the general information provided in Configuring your system for database coordination.

Warning: If you run db2profile on UNIX platforms, the environment variable LIBPATH and LD_LIBRARY_PATH are set. It is advisable to unset these environment variables, see appropriate *Quick Beginnings* Guide.

- Checking the DB2 environment variable settings
- Creating the DB2 switch load file
- Adding resource manager configuration information for DB2
- Changing DB2 configuration parameters

Parent topic: Scenario 1: Queue manager performs the coordination

This build: January 26, 2011 11:55:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13570_

6.4.2.2.1. Checking the DB2 environment variable settings

Ensure that your DB2® environment variables are set for queue manager processes *as well as in* your application processes. In particular, you must always set the DB2INSTANCE environment variable *before* you start the queue manager. The DB2INSTANCE environment variable identifies the DB2 instance containing the DB2 databases that are being updated. For example:

- On UNIX systems, use:

  ```
  export DB2INSTANCE=db2inst1
  ```

- On Windows systems, use:

  ```
  set DB2INSTANCE=DB2
  ```

Parent topic: DB2 configuration

This build: January 26, 2011 11:55:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13580_

6.4.2.2.2. Creating the DB2 switch load file

The easiest way to create the DB2® switch load file is to use the sample file xaswit.mak, which WebSphere® MQ provides to build the switch load files for a variety of database products.

On Windows systems, you can find xaswit.mak in the directory C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm. To create the DB2 switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak db2swit.dll
```

The generated switch file is placed in c:\Program Files\IBM\WebSphere MQ\exits.

On AIX®, you can find xaswit.mak in the directory /usr/mqm/samp/xatm; on other UNIX systems, you can find it in the directory /opt/mqm/samp/xatm.

Edit xaswit.mak to *uncomment* the lines appropriate to the version of DB2 you are using. Then execute the makefile using the command:

```
make -f xaswit.mak db2swit
```

The generated 32-bit switch load file is placed in /var/mqm/exits.

The generated 64-bit switch load file is placed in /var/mqm/exits64.

Parent topic: »DB2 configuration«

This build: January 26, 2011 11:55:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13590_

6.4.2.2.3. Adding resource manager configuration information for DB2

The next step is to modify the configuration information for the queue manager, as described in Adding configuration information to the queue manager, to declare DB2® as a participant in global units of work.

- On Windows and Linux (x86 platform) systems use the WebSphere® MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Figure 1 is a UNIX sample, showing an XAResourceManager entry where the database to be coordinated is called mydbname, this name being specified in the XAOpenString:

Figure 1. Sample XAResourceManager entry for DB2 on UNIX platforms

```
XAResourceManager:
   Name=mydb2
   SwitchFile=db2swit
   XAOpenString=mydbname,myuser,mypasswd,toc=t
   ThreadOfControl=THREAD
```

Note:

1. »ThreadOfControl=THREAD cannot be used with DB2 versions earlier than version 8. It is recommended that you always set ThreadOfControl and the XAOpenString parameter toc to one of the following combinations:
   - ThreadOfControl=THREAD and toc=t
   - ThreadOfControl=PROCESS and toc=p

   If you are using the jdbcdb2 XA switch load file to enable JDBC/JTA coordination, you must use ThreadOfControl=PROCESS and toc=p.
   «

Parent topic: »DB2 configuration«

This build: January 26, 2011 11:55:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13600_

6.4.2.2.4. Changing DB2 configuration parameters

For each DB2® database that the queue manager is coordinating, you need to:

Set database privileges

The queue manager processes run with effective user and group mqm on UNIX systems. On Windows systems, they run as the user that started the queue manager. This can be one of:

1. The user who issued the strmqm command, or
2. The user under which the IBM® MQSeries® Service COM server runs

By default, this user is called MUSR_MQADMIN.

If you have not specified a user name and password on the xa_open string, the user under which the queue manager is running is used by DB2 to authenticate the xa_open call. If

this user (for example, user mqm on UNIX systems) does not have minimal privileges in the database, the database refuses to authenticate the xa_open call.

The same considerations apply to your application process. If you have not specified a user name and password on the xa_open string, the user under which your application is running is used by DB2 to authenticate the xa_open call that is made during the first MQBEGIN. Again, this user must have minimal privileges in the database for this to work.

For example, give the mqm user connect authority in the mydbname database by issuing the following DB2 commands:

```
db2 connect to mydbname
db2 grant connect on database to user mqm
```

See Security considerations for more information about security.

Change the tp_mon_name parameter

For DB2 for Windows systems only, change the TP_MON_NAME configuration parameter to name the DLL that DB2 uses to call the queue manager for dynamic registration.

Use the command db2 update dbm cfg using TP_MON_NAME mqmax to name MQMAX.DLL as the library that DB2 uses to call the queue manager. This must be present in a directory within PATH.

Reset the maxappls parameter

You might need to review your setting for the *maxappls* parameter, which limits the maximum number of applications that can be connected to a database. Refer to Database connections.

Parent topic: »DB2 configuration«

This build: January 26, 2011 11:55:03

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13610_

6.4.2.3. Oracle configuration

Do the following:

1. Check environment variable settings.
2. Create the Oracle switch load file.
3. Add resource manager configuration information.
4. Change the Oracle configuration parameters, if necessary.

A current list of levels of Oracle supported by WebSphere® MQ is provided on the WebSphere MQ detailed system requirements page.

»Checking the Oracle environment variable settings«
Ensure that your Oracle environment variables are set for queue manager processes and in your application processes.

»Creating the Oracle switch load file«

»Adding resource manager configuration information for Oracle«

»Changing Oracle configuration parameters«

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:03

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13620_

6.4.2.3.1. Checking the Oracle environment variable settings

Ensure that your Oracle environment variables are set for queue manager processes and in your application processes.

In particular, always set the following environment variables before starting the queue manager:

ORACLE_HOME

The Oracle home directory. For example, on UNIX and Linux systems, use:

```
export ORACLE_HOME=/opt/oracle/product/8.1.6
```

On Windows systems, use:

```
set ORACLE_HOME=c:\oracle\ora81
```

ORACLE_SID

The Oracle SID being used. If you are using Net8 for client/server connectivity, you might not have to set this environment variable. Consult your Oracle documentation.

The subsequent example is an example of setting this environment variable on UNIX and Linux systems:

```
export ORACLE_SID=sid1
```

The equivalent on Windows systems is:

```
set ORACLE_SID=sid1
```

If you run queue managers on Windows 64 bit systems, then both 64 bit and 32 bit Oracle clients must be installed. You must install both clients because the queue manager runs as 32 bit processes that use a 32 bit switch load file, which must in turn start a 32 bit Oracle client dll.

The switch load file, loaded by 64 bit queue managers, must access the Oracle 64 bit client libraries. 32 bit queue managers must access the 32 bit Oracle client when WebSphere® MQ is running on a Windows 64 bit system.

Parent topic: »Oracle configuration«

This build: January 26, 2011 11:55:03

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13630_

6.4.2.3.2. Creating the Oracle switch load file

The easiest way to create the Oracle switch load file is to use the sample file xaswit.mak, which WebSphere® MQ provides to build the switch load files for a variety of database products.

On Windows systems, you can find xaswit.mak in the directory C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm. To create the Oracle switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak oraswit.dll
```

The generated switch file is placed in c:\Program Files\IBM\WebSphere MQ\exits.

On AIX®, you can find xaswit.mak in the directory /usr/mqm/samp/xatm; on other UNIX systems, you can find it in the directory /opt/mqm/samp/xatm.

Edit xaswit.mak to *uncomment* the lines appropriate to the version of Oracle you are using. Then execute the makefile using the command:

```
make -f xaswit.mak oraswit
```

The generated 32-bit switch load file is placed in /var/mqm/exits.

The generated 64-bit switch load file is placed in /var/mqm/exits64.

Parent topic: »Oracle configuration«

This build: January 26, 2011 11:55:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13640_

6.4.2.3.3. Adding resource manager configuration information for Oracle

The next step is to modify the configuration information for the queue manager, as described

in Adding configuration information to the queue manager, to declare Oracle as a participant in global units of work.

- On Windows and Linux (x86 platform) systems use the WebSphere® MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Figure 1 is a UNIX sample showing an XAResourceManager entry. It is recommend that you add a LogDir to the XA open string so that all error and tracing information is logged to the same place.

Figure 1. Sample XAResourceManager entry for Oracle on UNIX platforms

```
XAResourceManager:
  Name=myoracle
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypasswd+SesTm=35+LogDir=/tmp+threads=true
  ThreadOfControl=THREAD
```

Note:

1. In Figure 1, the xa_open string has been used with four parameters. Additional parameters can be included as described in Oracle's documentation.
2. When using Oracle with WebSphere MQ V5.3 for AIX®, you must add the +SqlNet clause to the xa_open string. WebSphere MQ V5.3 for AIX, uses AIX extended shared memory facilities and the +SqlNet client/server connectivity solution must be used to enable Oracle's client library to reach the Oracle server. Additional text must be added to the Oracle listener and tnsnames files, see Oracle's XA and Net9 documentation for details.
3. When using the WebSphere MQ parameter ThreadOfControl=THREAD you must use the Oracle parameter +threads=true in the XAResourceManager stanza.

See the *Oracle8 Server Application Developer's Guide* for more information on the xa_open string.

Parent topic: »Oracle configuration«

This build: January 26, 2011 11:55:04

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13650_

6.4.2.3.4. Changing Oracle configuration parameters

For each Oracle database that the queue manager is coordinating, you need to:

Review your maximum sessions

You might need to review your LICENSE_MAX_SESSIONS and PROCESSES settings to take into account the additional connections required by processes belonging to the queue manager. Refer to Database connections for more details.

Set database privileges

The Oracle user name specified in the xa_open string must have privileges to access the DBA_PENDING_TRANSACTIONS view, as described in the Oracle documentation.

The necessary privilege can be given using the following example command:

```
grant select on DBA_PENDING_TRANSACTIONS to myuser;
```

Parent topic: »Oracle configuration«

This build: January 26, 2011 11:55:04

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13660_

6.4.2.4. Informix configuration

Do the following:

1. Ensure that you have installed the appropriate Informix® client SDK:

- 32-bit queue managers and applications require a 32-bit Informix client SDK.
- 64-bit queue managers and applications require a 64-bit Informix client SDK.

2. Ensure Informix databases are created correctly.
3. Check environment variable settings.
4. Build the Informix switch load file.
5. Add resource manager configuration information.

A current list of levels of Informix supported by WebSphere® MQ is provided on the WebSphere MQ detailed system requirements page.

»Ensuring Informix databases are created correctly«

»Checking the Informix environment variable settings«

»Creating the Informix switch load file«

»Adding resource manager configuration information for Informix«

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13670_

6.4.2.4.1. Ensuring Informix databases are created correctly

Every Informix® database that is to be coordinated by a WebSphere® MQ queue manager must be created specifying the log parameter. For example:

```
create database mydbname with log;
```

WebSphere MQ queue managers are unable to coordinate Informix databases that do not have the log parameter specified on creation. If a queue manager attempts to coordinate an Informix database that does not have the log parameter specified on creation, the xa_open call to Informix will fail, and a number of FFST™ errors will be generated.

Parent topic: »Informix configuration«

This build: January 26, 2011 11:55:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13680_

6.4.2.4.2. Checking the Informix environment variable settings

Ensure that your Informix® environment variables are set for queue manager processes *as well as in* your application processes. In particular, always set the following environment variables before starting the queue manager:

INFORMIXDIR

The directory of the Informix product installation.

- For 32-bit UNIX applications, use the following command:

  ```
  export INFORMIXDIR=/opt/informix/32-bit
  ```

- For 64-bit UNIX applications, use the following command:

  ```
  export INFORMIXDIR=/opt/informix/64-bit
  ```

- For Windows applications, use the following command:

  ```
  set INFORMIXDIR=c:\informix
  ```

For systems that have 64-bit queue managers that must support both 32-bit and 64-bit applications, you need both the Informix 32-bit and 64-bit client SDKs installed. The sample make file xaswit.mak, used for creating a switch load file also sets both product installation directories.

INFORMIXSERVER

The name of the Informix server. For example, on UNIX systems, use:

```
export INFORMIXSERVER=hostname_1
```

On Windows systems, use:

```
set INFORMIXSERVER=hostname_1
```

ONCONFIG

The name of the Informix server configuration file. For example, on UNIX systems, use:

```
export ONCONFIG=onconfig.hostname_1
```

On Windows systems, use:

```
set ONCONFIG=onconfig.hostname_1
```

Parent topic: »Informix configuration«

This build: January 26, 2011 11:55:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13690_

6.4.2.4.3. Creating the Informix switch load file

The easiest way to create the Informix® switch load file is to use the sample file xaswit.mak, which WebSphere® MQ provides to build the switch load files for a variety of database products.

On Windows systems, you can find xaswit.mak in the directory C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm. To create the Informix switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak infswit.dll
```

The generated switch file is placed in c:\Program Files\IBM\WebSphere MQ\exits.

On AIX®, you can find xaswit.mak in the directory /usr/mqm/samp/xatm; on other UNIX systems, you can find it in the directory /opt/mqm/samp/xatm.

Edit xaswit.mak to *uncomment* the lines appropriate to the version of Informix you are using. Then execute the makefile using the command:

```
make -f xaswit.mak infswit
```

The generated 32-bit switch load file is placed in /var/mqm/exits.

The generated 64-bit switch load file is placed in /var/mqm/exits64.

Parent topic: »Informix configuration«

This build: January 26, 2011 11:55:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13700_

6.4.2.4.4. Adding resource manager configuration information for Informix

The next step is to modify the configuration information for the queue manager, as described in Adding configuration information to the queue manager, to declare Informix® as a participant in global units of work.

- On Windows and Linux (x86 platform) systems, use the WebSphere® MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the qm.ini file of the queue manager.

»Figure 1 is a UNIX sample, showing a qm.ini XAResourceManager entry where the database to be coordinated is called mydbname, this name being specified in the XAOpenString:
Figure 1. Sample XAResourceManager entry for Informix on UNIX platforms«

```
XAResourceManager:
  Name=myinformix
```

```
SwitchFile=infswit
XAOpenString=mydbname@myinformixserver
ThreadOfControl=THREAD
```

Note: By default the sample xaswit.mak on UNIX platforms creates a switch load file that uses threaded Informix libraries. You must ensure that ThreadOfControl is set to THREAD when using these Informix libraries. In Figure 1, the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to THREAD. When THREAD is specified, applications must be built using the threaded Informix libraries and the WebSphere MQ threaded API libraries.

The XAOpenString attribute must contain the database name, followed by the @ symbol, and then followed by the Informix server name.

To use the nonthreaded Informix libraries, you must ensure that the qm.ini file XAResourceManager stanza attribute ThreadOfControl is set to PROCESS. You must also make the following changes to the sample xaswit.mak:

1. Uncomment the generation of a nonthreaded switch load file.
2. Comment out the generation of the threaded switch load file.

Parent topic: Informix configuration

This build: January 26, 2011 11:55:04

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13710_

6.4.2.5. Sybase configuration

Do the following:

1. Ensure you have installed the Sybase XA libraries, for example by installing the XA DTM option.
2. Check environment variable settings.
3. Enable Sybase XA support.
4. Create the Sybase switch load file.
5. Add resource manager configuration information.

A current list of levels of Sybase supported by WebSphere® MQ is provided on the WebSphere MQ detailed system requirements page.

Checking the Sybase environment variable settings

Enabling Sybase XA support

Creating the Sybase switch load file

Adding resource manager configuration information for Sybase

Using multi-threaded programs with Sybase

Parent topic: Scenario 1: Queue manager performs the coordination

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13720_

6.4.2.5.1. Checking the Sybase environment variable settings

Ensure that your Sybase environment variables are set for queue manager processes *as well as in* your application processes. In particular, always set the following environment variables before starting the queue manager:

SYBASE

The location of the Sybase product installation. For example, on UNIX systems, use:

```
export SYBASE=/sybase
```

On Windows systems, use:

```
set SYBASE=c:\sybase
```

SYBASE_OCS

The directory under SYBASE where you have installed the Sybase client files. For example, on UNIX systems, use:

```
export SYBASE_OCS=OCS-12_0
```

On Windows systems, use:

```
set SYBASE_OCS=OCS-12_0
```

Parent topic: »Sybase configuration«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13730_

6.4.2.5.2. Enabling Sybase XA support

Within the Sybase XA configuration file ($SYBASE/$SYBASE_OCS/xa_config), define a Logical Resource Manager (LRM) for each connection to the Sybase server that is being updated.

An example of the contents of $SYBASE/$SYBASE_OCS/xa_config is shown in Figure 1.

Figure 1. Example contents of $SYBASE/$SYBASE_OCS/xa_config

```
# The first line must always be a comment

[xa]

  LRM=lrmname
  server=servername
```

Parent topic: »Sybase configuration«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13740_

6.4.2.5.3. Creating the Sybase switch load file

The easiest way to create the Sybase switch load file is to use the sample files supplied with WebSphere® MQ.

On Windows systems, you can find xaswit.mak in the directory C:\Program Files\IBM\WebSphere MQ\tools\c\samples\xatm. To create the Sybase switch load file with Microsoft Visual C++, use:

```
nmake /f xaswit.mak sybswit.dll
```

The generated switch file is placed in c:\Program Files\IBM\WebSphere MQ\exits.

On AIX®, you can find xaswit.mak in the directory /usr/mqm/samp/xatm; on other UNIX systems, you can find it in the directory /opt/mqm/samp/xatm.

Edit xaswit.mak to *uncomment* the lines appropriate to the version of Sybase you are using. Then execute the makefile using the command:

```
make -f xaswit.mak sybswit
```

The generated 32-bit switch load file is placed in /var/mqm/exits.

The generated 64-bit switch load file is placed in /var/mqm/exits64.

Parent topic: »Sybase configuration«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13750_

6.4.2.5.4. Adding resource manager configuration information for Sybase

The next step is to modify the configuration information for the queue manager, as described in Adding configuration information to the queue manager, to declare Sybase as a participant in global units of work.

- On Windows and Linux (x86 platform) systems use the WebSphere® MQ Explorer. Specify the details of the switch load file in the queue manager properties panel, under XA resource manager.
- On all other systems specify the details of the switch load file in the XAResourceManager stanza in the queue manager's qm.ini file.

Figure 1 shows a UNIX sample, which uses the database associated with the *lrmname* LRM definition in the Sybase XA configuration file, $SYBASE/$SYBASE_OCS/xa_config. Include a log file name if you want XA function calls to be logged:

Figure 1. Sample XAResourceManager entry for Sybase on UNIX platforms

```
XAResourceManager:
  Name=mysybase
  SwitchFile=sybswit
  XAOpenString=-Uuser -Ppassword -Nlrmname -L/tmp/sybase.log -Txa
  ThreadOfControl=THREAD
```

Parent topic: »Sybase configuration«

This build: January 26, 2011 11:55:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13760_

6.4.2.5.5. Using multi-threaded programs with Sybase

If you are using multi-threaded programs with WebSphere® MQ global units of work incorporating updates to Sybase, you *must* use the value THREAD for the ThreadOfControl parameter as shown in Figure 1.

Also ensure that you link your program (and the switch load file) with the threadsafe Sybase libraries (the _r versions).

Parent topic: »Sybase configuration«

This build: January 26, 2011 11:55:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13770_

6.4.2.6. Multiple database configurations

If you want to configure the queue manager so that updates to multiple databases can be included within global units of work, add an XAResourceManager stanza for each database.

If the databases are all managed by the same database manager, each stanza defines a separate database. Each stanza specifies the same *SwitchFile*, but the contents of the *XAOpenString* are different because it specifies the name of the database being updated. For example, the stanzas shown in Figure 1 configure the queue manager with the DB2® databases *MQBankDB* and *MQFeeDB* on UNIX systems.

Figure 1. Sample XAResourceManager entries for multiple DB2 databases

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=DB2 MQFeeDB
  SwitchFile=db2swit
```

```
    XAOpenString=MQFeeDB
```

If the databases to be updated are managed by different database managers, add an XAResourceManager stanza for each. In this case, each stanza specifies a different *SwitchFile*. For example, if *MQFeeDB* is managed by Oracle instead of DB2, use the following stanzas on UNIX systems:

Figure 2. Sample XAResourceManager entries for a DB2 and Oracle database

```
XAResourceManager:
  Name=DB2 MQBankDB
  SwitchFile=db2swit
  XAOpenString=MQBankDB

XAResourceManager:
  Name=Oracle MQFeeDB
  SwitchFile=oraswit
  XAOpenString=Oracle_XA+Acc=P/myuser/mypassword+SesTm=35+LogDir=/tmp/ora.log+DB=MQFeeDB
```

In principle, there is no limit to the number of database instances that can be configured with a single queue manager.

Note: For information on support for including Informix® databases in multiple database updates within global units of work, check the product readme file.

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13780_

6.4.2.7. Security considerations

The following information is provided for guidance only. In all cases, refer to the documentation provided with the database manager to determine the security implications of running your database under the XA model.

An application process denotes the start of a global unit of work using the MQBEGIN verb. The first MQBEGIN call that an application issues connects to all participating databases by calling their client library code at the xa_open entry point. All the database managers provide a mechanism for supplying a user ID and password in their XAOpenString. This is the only time that authentication information flows.

Note that, on UNIX platforms, fastpath applications must run with an effective user ID of mqm while making MQI calls.

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13790_

6.4.2.8. Administration tasks

In normal operations, only a minimal amount of administration is necessary after you have completed the configuration steps. The administration job is made easier because the queue manager tolerates database managers not being available. In particular this means that:

- The queue manager can start at any time without first starting each of the database managers.
- The queue manager does not need to stop and restart if one of the database managers becomes unavailable.

This allows you to start and stop the queue manager independently from the database server.

Whenever contact is lost between the queue manager and a database, they need to resynchronize when both become available again. Resynchronization is the process by which any in-doubt units of work involving that database are completed. In general, this occurs automatically without the need for user intervention. The queue manager asks the database for a list of units of work that are in doubt. It then instructs the database to either

commit or roll back each of these in-doubt units of work.

When a queue manager starts, it resynchronizes with each database. When an individual database becomes unavailable, only that database needs to be resynchronized the next time that the queue manager notices it is available again.

The queue manager regains contact with a previously unavailable database automatically as new global units of work are started with MQBEGIN. It does this by calling the xa_open function in the database client library. If this xa_open call fails, MQBEGIN returns with a completion code of MQCC_WARNING and a reason code of MQRC_PARTICIPANT_NOT_AVAILABLE. You can retry the MQBEGIN call later.

Do not continue to attempt a global unit of work that involves updates to a database that has indicated failure during MQBEGIN. There will not be a connection to that database through which updates can be made. Your only options are to end the program, or to retry MQBEGIN periodically in the hope that the database might become available again.

Alternatively, you can use the rsvmqtrn command to resolve explicitly all in-doubt units of work.

- In-doubt units of work
- Displaying outstanding units of work with the dspmqtrn command
- Resolving outstanding units of work with the rsvmqtrn command
- Mixed outcomes and errors
- Changing configuration information

Parent topic: Scenario 1: Queue manager performs the coordination

This build: January 26, 2011 11:55:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13800_

6.4.2.8.1. In-doubt units of work

A database might be left with in-doubt units of work if contact with the queue manager is lost after the database manager has been instructed to prepare. Until the database server receives the outcome from the queue manager (commit or roll back), it needs to retain the database locks associated with the updates.

Because these locks prevent other applications from updating or reading database records, resynchronization needs to take place as soon as possible.

If, for some reason, you cannot wait for the queue manager to resynchronize with the database automatically, you can use facilities provided by the database manager to commit or roll back the database updates manually. In the *X/Open Distributed Transaction Processing: The XA Specification*, this is called making a *heuristic* decision. Use it only as a last resort because of the possibility of compromising data integrity; you might, for example, mistakenly roll back the database updates when all other participants have committed their updates.

It is far better to restart the queue manager, or use the rsvmqtrn command when the database has been restarted, to initiate automatic resynchronization.

Parent topic: Administration tasks

This build: January 26, 2011 11:55:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13810_

6.4.2.8.2. Displaying outstanding units of work with the dspmqtrn command

While a database manager is unavailable, you can use the dspmqtrn command to check the state of outstanding global units of work involving that database.

The dspmqtrn command displays only those units of work in which one or more participants are in doubt, awaiting the decision from the queue manager to commit or roll back the prepared updates.

For each of these global units of work, the state of each participant is displayed in the output from dspmqtrn. If the unit of work did not update the resources of a particular resource manager, it is not displayed.

With respect to an in-doubt unit of work, a resource manager is said to have done one of the following things:

Prepared

The resource manager is prepared to commit its updates.

Committed

The resource manager has committed its updates.

Rolled-back

The resource manager has rolled back its updates.

Participated

The resource manager is a participant, but has not prepared, committed, or rolled back its updates.

When the queue manager is restarted, it asks each database having an XAResourceManager stanza for a list of its in-doubt global units of work. If the database has not been restarted, or is otherwise unavailable, the queue manager cannot yet deliver to the database the final outcomes for those units of work. The outcome of the in-doubt units of work is delivered to the database at the first opportunity when the database is again available.

In this case, the database manager is reported as being in *prepared* state until such time as resynchronization has occurred.

Whenever the dspmqtrn command displays an in-doubt unit of work, it first lists all the possible resource managers that could be participating. These are allocated a unique identifier, *RMId*, which is used instead of the *Name* of the resource managers when reporting their state with respect to an in-doubt unit of work.

Figure 1 shows the result of issuing the following command:

```
dspmqtrn -m MY_QMGR
```

Figure 1. Sample dspmqtrn output

```
AMQ7107: Resource manager 0 is MQSeries.
AMQ7107: Resource manager 1 is DB2 MQBankDB.
AMQ7107: Resource manager 2 is DB2 MQFeeDB.

AMQ7056: Transaction number 0,1.
     XID: formatID 5067085, gtrid_length 12, bqual_length 4
          gtrid [3291A50600002013746573 74]
          bqual [00000001]
AMQ7105: Resource manager 0 has committed.
AMQ7104: Resource manager 1 has prepared.
AMQ7104: Resource manager 2 has prepared.
```

where *Transaction number* is the ID of the transaction which can be used with the rsvmqtrn command. See AMQ7000-7999: WebSphere MQ product for further information about AMQ7056 message. The *XID* variables are part of the *X/Open XA Specification*; for the most up to date information on this specification see: http://www.opengroup.org/publications/catalog/c193.htm.

The output in Figure 1 shows that there are three resource managers associated with the queue manager. The first is resource manager 0, which is the queue manager itself. The other two resource manager instances are the MQBankDB and MQFeeDB DB2® databases.

The example shows only a single in-doubt unit of work. A message is issued for all three resource managers, which means that updates were made to the queue manager and both DB2 databases within the unit of work.

The updates made to the queue manager, resource manager 0, have been *committed*. The updates to the DB2 databases are in *prepared* state, which means that DB2 must have become unavailable before it was called to commit the updates to the *MQBankDB* and *MQFeeDB* databases.

The in-doubt unit of work has an external identifier called an XID (*transaction id*). This is a piece of data given to DB2 by the queue manager to identify its portion of the global unit of work.

Parent topic: »Administration tasks«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13820_

6.4.2.8.3. Resolving outstanding units of work with the rsvmqtrn command

The output shown in Figure 1 shows a single in-doubt unit of work in which the commit decision has yet to be delivered to both DB2® databases.

To complete this unit of work, the queue manager and DB2 need to resynchronize when DB2 next becomes available. The queue manager uses the start of new units of work as an opportunity to regain contact with DB2. Alternatively, you can instruct the queue manager to resynchronize explicitly using the rsvmqtrn command.

Do this soon after DB2 has been restarted, so that any database locks associated with the in-doubt unit of work are released as quickly as possible. Use the -a option, which tells the queue manager to resolve all in-doubt units of work. In the following example, DB2 has restarted, so the queue manager can resolve the in-doubt unit of work:

```
> rsvmqtrn -m MY_QMGR -a
Any in-doubt transactions have been resolved.
```

Parent topic: »Administration tasks«

This build: January 26, 2011 11:55:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13830_

6.4.2.8.4. Mixed outcomes and errors

Although the queue manager uses a two-phase commit protocol, this does not completely remove the possibility of some units of work completing with mixed outcomes. This is where some participants commit their updates and some back out their updates.

Units of work that complete with a mixed outcome have serious implications because shared resources that should have been updated as a single unit of work are no longer in a consistent state.

Mixed outcomes are mainly caused when heuristic decisions are made about units of work instead of allowing the queue manager to resolve in-doubt units of work itself. Such decisions are outside the queue manager's control.

Whenever the queue manager detects a mixed outcome, it produces FFST™ information and documents the failure in its error logs, with one of two messages:

- If a database manager rolls back instead of committing:

  ```
  AMQ7606 A transaction has been committed but one or more resource
          managers have rolled back.
  ```

- If a database manager commits instead of rolling back:

  ```
  AMQ7607 A transaction has been rolled back but one or more resource
          managers have committed.
  ```

Further messages identify the databases that are heuristically damaged. It is then your responsibility to locally restore consistency to the affected databases. This is a complicated procedure in which you need first to isolate the update that has been wrongly committed or rolled back, then to undo or redo the database change manually.

Parent topic: »Administration tasks«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13840_

6.4.2.8.5. Changing configuration information

After the queue manager has successfully started to coordinate global units of work, do not change any of the resource manager configuration information.

If you need to change the configuration information you can do so at any time, but the changes do not take effect until after the queue manager has been restarted.

If you remove the resource manager configuration information for a database, you are effectively removing the ability for the queue manager to contact that database manager.

Never change the *Name* attribute in any of your resource manager configuration information. This attribute uniquely identifies that database manager instance to the queue manager. If you change this unique identifier, the queue manager assumes that the database has been removed and a completely new instance has been added. The queue manager still associates outstanding units of work with the old *Name*, possibly leaving the database in an in-doubt state.

»Removing database manager instances«

Parent topic: »Administration tasks«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13850_

6.4.2.8.5.1. Removing database manager instances

If you need to remove a database from your configuration permanently, ensure that the database is not in doubt before you restart the queue manager. Database products provide commands for listing in-doubt transactions. If there are any in-doubt transactions, first allow the queue manager to resynchronize with the database. Do this by starting the queue manager. You can verify that resynchronization has taken place by using the rsvmqtrn command or the database's own command for viewing in-doubt units of work. Once you are satisfied that resynchronization has taken place, end the queue manager and remove the database's configuration information.

If you fail to observe this procedure the queue manager still remembers all in-doubt units of work involving that database. A warning message, AMQ7623, is issued every time the queue manager is restarted. If you are never going to configure this database with the queue manager again, use the -r option of the rsvmqtrn command to instruct the queue manager to forget about the database's participation in its in-doubt transactions. The queue manager forgets about such transactions only when in-doubt transactions have been completed with all participants.

There are times when you might need to remove some resource manager configuration information temporarily. On UNIX systems this is best achieved by commenting out the stanza so that it can be easily reinstated at a later time. You might decide to do this if there are errors every time the queue manager contacts a particular database or database manager. Temporarily removing the resource manager configuration information concerned allows the queue manager to start global units of work involving all the other participants. Here is an example of a commented-out XAResourceManager stanza follows:
Figure 1. Commented- out XAResourceManager stanza on UNIX systems

```
# This database has been temporarily removed
#XAResourceManager:
#   Name=mydb2
#   SwitchFile=db2swit
#   XAOpenString=mydbname,myuser,mypassword,toc=t
#   ThreadOfControl=THREAD
```

On Windows systems, use the WebSphere® MQExplorer to delete the information about the database manager instance. Take great care to type in the correct name in the *Name* field when reinstating it. If you mistype the name, you may face in-doubt problems, as described

in Changing configuration information.

Parent topic: »Changing configuration information«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13860_

6.4.2.9. XA dynamic registration

The XA specification provides a way of reducing the number of xa_* calls that a transaction manager makes to a resource manager. This optimization is known as *dynamic registration*. Dynamic registration is supported by DB2®. Other databases might support it; consult the documentation for your database product for details.

Why is the dynamic registration optimization useful? In your application, some global units of work might contain updates to database tables; others might not contain such updates. When no persistent update has been made to a database's tables, there is no need to include that database in the commit protocol that occurs during MQCMIT.

Whether or not your database supports dynamic registration, your application calls xa_open during the first MQBEGIN call on a WebSphere® MQ connection. It also calls xa_close on the subsequent MQDISC call. The pattern of subsequent XA calls depends on whether the database supports dynamic registration:

If your database does not support dynamic registration...

Every global unit of work involves several XA function calls made by WebSphere MQ code into the database client library, regardless of whether you made a persistent update to the tables of that database within your unit of work. These include:

- xa_start and xa_end from the application process. These are used to declare the beginning and end of a global unit of work.
- xa_prepare, xa_commit, and xa_rollback from the queue manager agent process, amqzlaa0. These are used to deliver the outcome of the global unit of work: the commit or rollback decision.

In addition, the queue manager agent process also calls xa_open during the first MQBEGIN.

If your database supports dynamic registration...

The WebSphere MQ code makes only those XA function calls that are necessary. For a global unit of work that has not involved persistent updates to database resources, there are no XA calls to the database. For a global unit of work that has involved such persistent updates, the calls are to:

- xa_end from the application process to declare the end of the global unit of work.
- xa_prepare, xa_commit, and xa_rollback from the queue manager agent process, amqzlaa0. These are used to deliver the outcome of the global unit of work: the commit or rollback decision.

For dynamic registration to work, it is vital that the database has a way of telling WebSphere MQ when it has performed a persistent update that it wants to be included in the current global unit of work. WebSphere MQ provides the ax_reg function for this purpose.

The database's client code that runs in your application process finds the ax_reg function and calls it, to *dynamically register* the fact it has done persistent work within the current global unit of work. In response to this ax_reg call, WebSphere MQ records that the database has participated. If this is the first ax_reg call on this WebSphere MQ connection, the queue manager agent process calls xa_open.

The database client code make this ax_reg call when it is running in your process, for example, during an SQL UPDATE call or whatever call in the database's client API is responsible

»Error conditions«

»Summarizing XA calls«

Parent topic: »Scenario 1: Queue manager performs the coordination«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13870_

6.4.2.9.1. Error conditions

There is an opportunity here for a confusing failure in the queue manager. Here is a common example. If you forget to set your database environment variables properly before starting your queue manager, the queue manager's calls to xa_open fail. No global units of work can be used.

To avoid this, ensure that you have set the relevant environment variables before starting the queue manager. Review your database product's documentation, and the advice given in Checking the DB2 environment variable settings, Checking the Oracle environment variable settings, and Checking the Sybase environment variable settings.

With all database products, the queue manager calls xa_open once at queue manager startup, as part of the recovery session (as explained in Administration tasks). This xa_open call fails if you set your database environment variables incorrectly, but it does not cause the queue manager to fail to start. This is because the same xa_open error code is used by the database client library to indicate that the database server is unavailable. We do not treat this as a serious error, as the queue manager must be able to start to continue processing data outside global units of work involving that database.

Subsequent calls to xa_open are made from the queue manager during the first MQBEGIN on a WebSphere® MQ connection (if dynamic registration is not being used) or during a call by the database client code to the WebSphere MQ-provided ax_reg function (if dynamic registration is being used).

The timing of any error conditions (or, occasionally, FFST™ reports) depends on whether you are using dynamic registration:

- If you are using dynamic registration, your MQBEGIN call could succeed, but your SQL UPDATE (or similar) database call will fail.
- If you are not using dynamic registration, your MQBEGIN call will fail.

Ensure that your environment variables are set correctly in your application and queue manager processes.
Parent topic: »XA dynamic registration«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13880_

6.4.2.9.2. Summarizing XA calls

Table 1 lists the calls that are made to the XA functions in a database client library as a result of the various MQI calls that control global units of work. This is not a complete description of the protocol described in the XA specification; we have provided it as a brief overview.

Note that xa_start and xa_end calls are always called by WebSphere® MQ code in the application process, whereas xa_prepare, xa_commit, and xa_rollback are always called from the queue manager agent process, amqzlaa0.

The xa_open and xa_close calls shown in this table are all made from the application process. The queue manager agent process calls xa_open in the circumstances described in Error conditions.

Table 1. Summary of XA function calls

MQI call	XA calls made with dynamic registration	XA calls made without dynamic registration
First MQBEGIN	xa_open	xa_open xa_start
Subsequent MQBEGIN	No XA calls	xa_start
MQCMIT (**without** ax_reg being	No XA calls	xa_end

called during the current global unit of work)		xa_prepare xa_commit xa_rollback
MQCMIT (**with** ax_reg being called during the current global unit of work)	xa_end xa_prepare xa_commit xa_rollback	Not applicable. No calls are made to ax_reg in non-dynamic mode.
MQBACK (**without** ax_reg being called during the current global unit of work)	No XA calls	xa_end xa_rollback
MQBACK (**with** ax_reg being called during the current global unit of work)	xa_end xa_rollback	Not applicable. No calls are made to ax_reg in non-dynamic mode.
MQDISC, where MQCMIT or MQBACK was called first. If they were not, MQCMIT processing is first done during MQDISC.	xa_close	xa_close

Notes:

1. For MQCMIT, xa_commit is called if xa_prepare is successful. Otherwise, xa_rollback is called.

Parent topic: »XA dynamic registration«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13890_

6.4.3. Scenario 2: Other software provides the coordination

In scenario 2, an external transaction manager coordinates global units of work, starting and committing them under control of the transaction manager's API. The MQBEGIN, MQCMIT, and MQBACK verbs are unavailable.

This section describes this scenario, including:

- External sync point coordination
- Using CICS
- Using the Microsoft Transaction Server (COM+)

»External sync point coordination«

»Using CICS«

»Using the Microsoft Transaction Server (COM+)«

Parent topic: »Transactional support«

This build: January 26, 2011 11:55:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13900_

6.4.3.1. External sync point coordination

A global unit of work can also be coordinated by an external X/Open XA-compliant transaction manager. Here the WebSphere® MQ queue manager participates in, but does not coordinate, the unit of work.

The flow of control in a global unit of work coordinated by an external transaction manager is as follows:

1. **An application tells the external sync point coordinator (for example, TXSeries) that it**

wants to start a transaction.

2. The sync point coordinator tells known resource managers, such as WebSphere MQ, about the current transaction.
3. The application issues calls to resource managers associated with the current transaction. For example, the application could issue MQGET calls to WebSphere MQ.
4. The application issues a commit or backout request to the external sync point coordinator.
5. The sync point coordinator completes the transaction by issuing the appropriate calls to each resource manager, typically using two-phase commit protocols.

The supported levels of external sync point coordinators that can provide a two-phase commit process for transactions in which WebSphere MQ participates are defined on the WebSphere MQ detailed system requirements page.

See the WebSphere MQ Application Programming Guide for information about writing and building transactions to be coordinated by an external sync point coordinator.

The rest of this chapter describes how to enable external units of work.

»The WebSphere MQ XA switch structure«

Parent topic: »Scenario 2: Other software provides the coordination«

This build: January 26, 2011 11:55:06

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13910_

6.4.3.1.1. The WebSphere MQ XA switch structure

Each resource manager participating in an externally coordinated unit of work must provide an XA switch structure. This structure defines both the capabilities of the resource manager and the functions that are to be called by the sync point coordinator.

WebSphere® MQ provides two versions of this structure:

- *MQRMIXASwitch* for static XA resource management
- *MQRMIXASwitchDynamic* for dynamic XA resource management

Consult your transaction manager documentation to determine whether to use the static or dynamic resource management interface. Wherever a transaction manager supports it, we recommend that you use dynamic XA resource management.

Some 64-bit transaction managers treat the *long* type in the XA specification as 64-bit, and some treat it as 32-bit. WebSphere MQ supports both models:

- If your transaction manager is 32-bit, or your transaction manager is 64-bit but treats the *long* type as 32-bit, use the switch load file listed in Table 1.
- If your transaction manager is 64-bit and treats the *long* type as 64-bit, use the switch load file listed in Table 2.

A list of known 64-bit transaction managers that treat the *long* type as 64-bit is provided in Table 3. Consult your transaction manager documentation if you are unsure which model your transaction manager uses.

Table 1. XA switch load file names

Platform	Switch load file name (server)	Switch load file name (extended transactional client)
Windows	*mqmxa.dll*	*mqcxa.dll*
AIX® (nonthreaded)	*libmqmxa.a*	*libmqcxa.a*
AIX (threaded)	*libmqmxa_r.a*	*libmqcxa_r.a*
HP-UX (nonthreaded)	*libmqmxa.so*	*libmqcxa.so*
HP-UX (threaded)	*libmqmxa_r.so*	*libmqcxa_r.so*
Linux (nonthreaded)	*libmqmxa.so*	*libmqcxa.so*
Linux (threaded)	*libmqmxa_r.so*	*libmqcxa_r.so*

Solaris	*libmqmxa.so*	*libmqcxa.so*

Table 2. Alternative 64-bit XA switch load file names

Platform	Switch load file name (server)	Switch load file name (extended transactional client)
AIX (nonthreaded)	*libmqmxa64.a*	*libmqcxa64.a*
AIX (threaded)	*libmqmxa64_r.a*	*libmqcxa64_r.a*
HP-UX (nonthreaded)	*libmqmxa64.so*	*libmqcxa64.so*
HP-UX (threaded)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Linux (nonthreaded)	*libmqmxa64.so*	*libmqcxa64.so*
Linux (threaded)	*libmqmxa64_r.so*	*libmqcxa64_r.so*
Solaris	*libmqmxa64.so*	*libmqcxa64.so*

Table 3. 64-bit transaction managers that require the alternate 64-bit switch load file

Transaction Manager
Tuxedo

Some external sync point coordinators (not CICS®) require that each resource manager participating in a unit of work supplies its name in the name field of the XA switch structure. The WebSphere MQ resource manager name is MQSeries_XA_RMI.

The sync point coordinator defines how the WebSphere MQ XA switch structure links to it. Information about linking the WebSphere MQ XA switch structure with CICS is provided in Using CICS. For information about linking the WebSphere MQ XA switch structure with other XA-compliant sync point coordinators, consult the documentation supplied with those products.

The following considerations apply to using WebSphere MQ with all XA-compliant sync point coordinators:

- **The xa_info structure passed on any xa_open call by the sync point coordinator includes the name of a WebSphere MQ queue manager. The name takes the same form as the queue-manager name passed to the MQCONN call. If the name passed on the xa_open call is blank, the default queue manager is used.**
  Alternatively, the xa_info structure can contain values for the *TPM* and *AXLIB* parameters. The *TPM* parameter specifies the transaction manager being used. The valid values are CICS, TUXEDO and ENCINA. The *AXLIB* parameter specifies the name of the library that contains the transaction manager's ax_reg and ax_unreg functions. For more information on these parameters, see the section about configuring extended transactional clients in WebSphere MQ Clients. If the xa_info structure contains either of these parameters, the queue manager name is specified in the *QMNAME* parameter, unless the default queue manager is being used.
- **Only one queue manager at a time can participate in a transaction coordinated by an instance of an external sync point coordinator. The sync point coordinator is effectively connected to the queue manager, and is subject to the rule that only one connection at a time is supported.**
- **All applications that include calls to an external sync point coordinator can connect only to the queue manager that is participating in the transaction managed by the external coordinator (because they are already effectively connected to that queue manager). However, such applications must issue an MQCONN call to obtain a connection handle, and an MQDISC call before they exit.**
- **A queue manager with resource updates coordinated by an external sync point coordinator must start before the external sync point coordinator. Similarly, the sync point coordinator must end before the queue manager.**
- **If your external sync point coordinator terminates abnormally, stop and restart your queue manager *before* restarting the sync point coordinator to ensure that any messaging operations uncommitted at the time of the failure are properly resolved.**

Parent topic: »External sync point coordination«

This build: January 26, 2011 11:55:07

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13920_

6.4.3.2. Using CICS

CICS® is one of the elements of TXSeries. The versions of TXSeries that are XA-compliant (and use a two-phase commit process) are defined on the WebSphere MQ detailed system requirements page.

- The CICS two-phase commit process

Parent topic: Scenario 2: Other software provides the coordination

This build: January 26, 2011 11:55:07

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13930_

6.4.3.2.1. The CICS® two-phase commit process

WebSphere® MQ also supports other transaction managers. See the WebSphere MQ detailed system requirements page for the current lists of supported software.

- Requirements of the two-phase commit process
- General XA support
- Enabling the CICS two-phase commit process
- Enabling CICS user exits

Parent topic: Using CICS

This build: January 26, 2011 11:55:07

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13940_

6.4.3.2.1.1. Requirements of the two-phase commit process

When you use the CICS® two-phase commit process with WebSphere® MQ, note the following requirements:

- WebSphere MQ and CICS must reside on the same physical machine.
- WebSphere MQ does not support CICS on a WebSphere MQ client.
- You must start the queue manager, with its name specified in the XAD resource definition stanza, *before* you attempt to start CICS. Failure to do this will prevent you from starting CICS if you have added an XAD resource definition stanza for WebSphere MQ to the CICS region.
- Only one WebSphere MQ queue manager can be accessed at a time from a single CICS region.
- A CICS transaction must issue an MQCONN request before it can access WebSphere MQ resources. The MQCONN call must specify the name of the WebSphere MQ queue manager specified on the XAOpen entry of the XAD resource definition stanza for the CICS region. If this entry is blank, the MQCONN request must specify the default queue manager.
- A CICS transaction that accesses WebSphere MQ resources must issue an MQDISC call from the transaction before returning to CICS. Failure to do this might mean that the CICS application server is still connected, leaving queues open. Additionally, if you do not install a task termination exit (see Sample task termination exit), the CICS application server might later end abnormally, perhaps during a subsequent transaction.
- You must ensure that the CICS user ID (cics) is a member of the mqm group, so that the CICS code has the authority to call WebSphere MQ.
  For transactions running in a CICS environment, the queue manager adapts its methods of authorization and determining context as follows:
  - The queue manager queries the user ID under which CICS runs the transaction. This is the user ID checked by the Object Authority Manager, and is used for context information.

- In the message context, the application type is MQAT_CICS.
- The application name in the context is copied from the CICS transaction name.

Parent topic: »The CICS two-phase commit process«

This build: January 26, 2011 11:55:07

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13950_

6.4.3.2.1.2. General XA support

General XA is not supported on i5/OS®.

An XA switch load module is provided to enable you to link CICS® with WebSphere® MQ on UNIX systems. Additionally, sample source code files are provided to enable you to develop the XA switches for other transaction messages. The names of the switch load modules provided are:

Table 1. Essential code for CICS applications: XA initialization routine

C (source)	C (exec) - add one of the following to your XAD.Stanza
amqzscix.c	amqzsc - TXSeries for AIX®, Version 5.1, amqzsc - TXSeries for HP-UX, Version 5.1 amqzsc - TXSeries for Sun Solaris, Version 5.1
amqzscin.c	mqmc4swi - TXSeries for Windows, Version 5.1

»Building libraries for use with TXSeries for Multiplatforms version 5«
Building libraries for use with TXSeries for Multiplatforms version 5

»Building libraries for use with TXSeries for Multiplatforms version 6«
Building libraries for use with TXSeries for Multiplatforms version 6

»WebSphere MQ XA support and Tuxedo«

Parent topic: »The CICS two-phase commit process«

This build: January 26, 2011 11:56:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fg13810_

6.4.3.2.1.2.1. Building libraries for use with TXSeries for Multiplatforms version 5

»Building libraries for use with TXSeries for Multiplatforms version 5«

Pre-built switch load files are supplied with WebSphere® MQ. These are shared libraries (DLLs on Windows) for use with CICS® programs in which you require 2-phase commit using the XA protocol. The names of these pre-built libraries are in the table Essential code for CICS applications: XA initialization routine. Sample source code is also supplied in the following directories:

Platform	Directory	Source file
AIX®	/usr/mqm/samp/	amqzscix.c
Other UNIX platforms	/opt/mqm/samp/	amqzscix.c
Windows	c:\Program Files\IBM\Websphere MQ\Tools\c\Samples	amqzscin.c

To build the switch load file from the sample source, follow the appropriate instructions for your platform:

AIX

Issue the following command:

```
xlC_r4 /usr/mqm/samp/amqzscix.c -I/usr/lpp/encina/include   -e amqzscix -o amqzsc
     /usr/lpp/cics/lib/regxa_swxa.o   -L/usr/lpp/cics/lib -L/usr/lpp/encina/lib
     -lmqmcics_r -lmqmxa_r   -lmqm_r -lcicsrt -lEncina -lEncServer -ldce
```

Solaris

Issue the following command:

```
/opt/SUNWspro/bin/cc -s -l/opt/encina/include amqzscix.c -G -o amqzscix  -e
     CICS_XA_Init -L/opt/encina/lib -L/opt/dcelocal/lib /opt/cics/lib/reqxa_swxa.o
     -lmqmcics -lmqmxa -lmqm -lmqmcs_d -lmqmzse -lcicsrt -lEncina -lEncSfs -ldce
```

HP-UX

Issue the following command:

```
cc -c -s -I/opt/encina/include /opt/mqm/samp/amqzscix.c -Aa +z -o  amqzscix.o ld -b
     -o amqzscix amqzscix.o /opt/cics/lib/regxa_swxa.o +e CICS_XA_Init \
     -L/opt/encina/lib -L/opt/cics/lib -lmqmxa_r -lmqm_r -ldbm -lc -lm
```

Windows

Follow these steps:

1. Use the cl command to build amqzscin.obj by compiling using at least the following:

   ```
   cl.exe -c -IEncinaPath\include -IWebSphereMQPath\include -Gz -LD amqzscin.c
   ```

2. Create a module definition file named mqmc1415.def containing the following lines:

   ```
   LIBRARY MQMC4SWI
   EXPORTS
   CICS_XA_Init
   ```

3. Use the lib command to build an export file and an import library using at least the following:

   ```
   lib -def:mqmc4swi.def -out:mqmc4swi.lib
   ```

   If the lib command is successful then an mqmc4swi.exp will also have been built.

4. Use the link command to build mqmc4swi.dll by linking using at least the following:

   ```
   link.exe -dll -nod -out:mqmc4swi.dll
        amqzscin.obj CicsPath\lib\regxa_swxa.obj
        mqmc4swi.exp mqmcics4.lib
       CicsPath\lib\libcicsrt.lib
       DcePath\lib\libdce.lib DcePath\lib\pthreads.lib
       EncinaPath\lib\libEncina.lib
       EncinaPath\lib\libEncServer.lib
       msvcrt.lib kernel32.lib
   ```

Parent topic: General XA support

This build: January 26, 2011 11:55:07

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13951_

6.4.3.2.1.2.2. Building libraries for use with TXSeries for Multiplatforms version 6

Building libraries for use with TXSeries for Multiplatforms version 6

The switch load file and task termination user exit are shared libraries (DLLs on Windows). The switch load file is needed with CICS programs in which you require 2-phase commit using the XA protocol. The task termination user exit is needed to provide clean shutdown of the MQ runtime in your application in case of system problems

The pre-built switch load file and task termination user exit supplied with WebSphere MQ are usable with TXSeries CICS version 5, but not with TXSeries version 6. It is necessary for you to build a new switch load file and task termination user exit before WebSphere MQ can be used with TXSeries version 6.

The sample source files supplied for use with TXSeries CICS version 5 (see the TXSeries CICS 5 page) can be used with TXSeries version 6, but please note you must compile and link them

using TXSeries 6 header files and libraries as per the requirements of TXSeries 6.

The names of these source files are the same as those listed on the TXSeries CICS 5 page. Consult the TXSeries sample makefiles and product documentation for examples of how the switch load file and task termination user exit must be compiled and linked.

Parent topic: »General XA support«

This build: January 26, 2011 11:55:07

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13952_

6.4.3.2.1.2.3. WebSphere MQ XA support and Tuxedo

WebSphere® MQ on Windows and UNIX systems can block Tuxedo-coordinated XA applications indefinitely in xa_start. This can occur only when two or more processes coordinated by Tuxedo in a single global transaction attempt to access WebSphere MQ using the same transaction branch ID (XID). If Tuxedo gives each process in the global transaction a different XID to use with WebSphere MQ, this cannot occur.

To avoid the problem, configure each application in Tuxedo that accesses WebSphere MQ under a single global transaction ID (gtrid), within its own Tuxedo server group. Processes in the same server group use the same XID when accessing resource managers on behalf of a single gtrid, and are therefore vulnerable to blocking in xa_start in WebSphere MQ. Processes in different server groups use separate XIDs when accessing resource managers and so do not have to serialize their transaction work in WebSphere MQ.

Parent topic: »General XA support«

This build: January 26, 2011 11:56:24

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fg13820_

6.4.3.2.1.3. Enabling the CICS two-phase commit process

To enable CICS® to use a two-phase commit process to coordinate transactions that include MQI calls, add a CICS XAD resource definition stanza entry to the CICS region.

Here is an example of adding an XAD stanza entry for WebSphere® MQ for Windows, where <Drive> is the drive where WebSphere MQ is installed (for example, D:).

```
cicsadd -cxad -r<cics_region> \
      ResourceDescription="MQM XA Product Description" \
      SwitchLoadFile="<Drive>:\Program Files\IBM\WebSphere MQ\bin\mqmc4swi.dll" \
      XAOpen=<queue_manager_name>
```

For extended transactional clients, use the switch load file mqcc4swi.dll.

Here is an example of adding an XAD stanza entry for WebSphere MQ for UNIX systems:

```
cicsadd -cxad -r<cics_region> \
       ResourceDescription="MQM XA Product Description" \
       SwitchLoadFile="/opt/mqm/lib/amqzsc" \
       XAOpen=<queue_manager_name>
```

For extended transactional clients, use the switch load file amqczsc.

For information about using the cicsadd command, see the *CICS Administration Reference*, or *CICS Administration Guide* for your platform.

Calls to WebSphere MQ can be included in a CICS transaction, and the WebSphere MQ resources will be committed or rolled back as directed by CICS. This support is not available to client applications.

You *must* issue an MQCONN from your CICS transaction in order to access WebSphere MQ resources, followed by a corresponding MQDISC on exit.

Parent topic: »The CICS two-phase commit process«

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13960_

6.4.3.2.1.4. Enabling CICS user exits

Before using a CICS® user exit, read the *CICS Administration Guide* for your platform.

A CICS user exit *point* (normally referred to as a *user exit*) is a place in a CICS module at which CICS can transfer control to a program that you have written (a user exit *program*), and at which CICS can resume control when your exit program has finished its work.

Sample task termination exit

Parent topic: The CICS two-phase commit process

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13970_

6.4.3.2.1.4.1. Sample task termination exit

WebSphere® MQ supplies sample source code for a CICS® task termination exit in the following directories:

Table 1. CICS task termination exits

Platform	Directory	Source file
AIX®	/opt/mqm/samp	amqzscgx.c
Other UNIX systems	/usr/mqm/samp	amqzscgx.c
Windows	c:\Program Files\IBM\Websphere MQ\Tools\c\Samples	amqzscgn.c

The build instructions for the sample task termination exit are contained in the comments near the top of each source file.

This exit is invoked by CICS at normal and abnormal task termination (after any sync point has been taken). No recoverable work is permitted in the exit program.

These functions are only used in a WebSphere MQ and CICS context in which the CICS version supports the XA interface. CICS refers to these libraries as "programs" or "user exits".

CICS has a number of user exits and amqzscgx, if used, is defined and enabled on CICS as the "Task termination user exit (UE014015)", that is, exit number 15.

When the task termination exit is called by CICS, CICS has already informed WebSphere MQ of the task's termination state and WebSphere MQ has taken the appropriate action (commit or rollback). All the exit does is to issue an MQDISC to clean up.

One purpose of installing and configuring your CICS system to use a task termination exit is to protect your system against some of the consequences of faulty application code. For example, if your CICS transaction ends abnormally without first calling MQDISC, and has no task termination exit installed, then you might see (within around 10 seconds) a subsequent catastrophic failure of the CICS region. This is because WebSphere MQ's health thread, that runs in the cicsas process, will not have been posted and given time to clean up and return. The symptoms might be that the cicsas process ends immediately, having written FFST™ reports to /var/mqm/errors or the equivalent location on Windows.

Parent topic: Enabling CICS user exits

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13971_

6.4.3.3. Using the Microsoft Transaction Server (COM+)

COM+ (Microsoft Transaction Server) is designed to help users run business logic applications in a typical middle tier server. COM+ divides work up into *activities*, which are typically short independent chunks of business logic, such as *transfer funds from account A to account B*. COM+ relies heavily on object orientation and in particular on COM; loosely a COM+ activity is represented by a COM (business) object.

COM+ is an integrated part of the operating system. To use COM+ on Windows 2000 and Windows XP, you need Hotfix Q313582 (also known as COM+ Rollup Package 19.1).

COM+ provides three services to the business object administrator, removing much of the worry from the business object programmer:

- Transaction management
- Security
- Resource pooling

You usually use COM+ with front end code that is a COM client to the objects held within COM+, and back end services such as a database, with WebSphere® MQ bridging between the COM+ business object and the back end.

The front end code can be a standalone program, or an Active Server Page (ASP) hosted by the Microsoft Internet Information Server (IIS). The front end code can reside on the same computer as COM+ and its business objects, with connection through COM. Alternatively, the front end code can reside on a different computer, with connection through DCOM. You can use different clients to access the same COM+ business object in different situations.

The back end code can reside on the same computer as COM+ and its business objects, or on a different computer with connection through any of the WebSphere MQ supported protocols.

For detailed information on using COM+, including how to configure it, and how to develop your applications and object code, read the *WebSphere MQ COM+ Component Services Support* part of the WebSphere MQ Help Center.

Parent topic: Scenario 2: Other software provides the coordination

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13980_

6.4.4. Unit of recovery disposition

WebSphere MQ for z/OS provides unit of recovery dispositions. This feature allows you to configure whether the second phase of 2-phase commit transactions can be driven, for example, during recovery, when connected to another queue manager within the same queue-sharing group (QSG).

WebSphere® MQ for z/OS V7.0.1 and later supports the unit of recovery disposition.

Unit of recovery disposition

The unit of recovery disposition is related to an application's connection and subsequently any transactions that it starts. There are two possible unit of recovery dispositions.

- A GROUP unit of recovery disposition identifies that a transactional application is logically connected to the queue-sharing group and does not have an affinity to any specific queue manager. Any 2-phase commit transactions that it starts that have completed phase-1 of the commit process, that is, they are in doubt, can be inquired and resolved, when connected to any queue manager within the QSG. In a recovery scenario this means that the transaction coordinator does not have to reconnect to the same queue manager, which might be unavailable.
- A QMGR unit of recovery disposition identifies that an application has a direct affinity to the queue manager to which it is connected and any transactions that it starts also have this disposition.
  In a recovery scenario the transaction coordinator must reconnect to the same queue manager to inquire, and resolve, any in-doubt transactions, irrespective of whether the queue manager belongs to a queue-sharing group.

See Unit of recovery disposition in a queue-sharing group for details how to implement this feature.

Parent topic: Transactional support

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa24000_

6.5. The WebSphere MQ dead-letter queue handler

What is a dead-letter queue, how are messages put on it, and how do you manage it?

A *dead-letter queue* (DLQ), sometimes referred to as an *undelivered-message queue*, is a holding queue for messages that cannot be delivered to their destination queues. Every queue manager in a network should have an associated DLQ.

Messages can be put on the DLQ by queue managers, message channel agents (MCAs), and applications. All messages on the DLQ must be prefixed with a *dead-letter header* structure, MQDLH.

Messages put on the DLQ by a queue manager or a message channel agent always have an MQDLH; applications putting messages on the DLQ must supply an MQDLH. The *Reason* field of the MQDLH structure contains a reason code that identifies why the message is on the DLQ.

All WebSphere® MQ environments need a routine to process messages on the DLQ regularly. WebSphere MQ supplies a default routine, called the *dead-letter queue handler* (the DLQ handler), which you invoke using the runmqdlq command.

Instructions for processing messages on the DLQ are supplied to the DLQ handler by means of a user-written *rules table*. That is, the DLQ handler matches messages on the DLQ against entries in the rules table; when a DLQ message matches an entry in the rules table, the DLQ handler performs the action associated with that entry.

Invoking the DLQ handler

The DLQ handler rules table

How the rules table is processed

An example DLQ handler rules table

Parent topic: Configuring and managing WebSphere MQ

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa13990_

6.5.1. Invoking the DLQ handler

Invoke the DLQ handler using the runmqdlq command. You can name the DLQ you want to process and the queue manager you want to use in two ways:

- As parameters to runmqdlq from the command prompt. For example:

  ```
  runmqdlq ABC1.DEAD.LETTER.QUEUE ABC1.QUEUE.MANAGER <qrule.rul
  ```

- In the rules table. For example:

  ```
  INPUTQ(ABC1.DEAD.LETTER.QUEUE) INPUTQM(ABC1.QUEUE.MANAGER)
  ```

The above examples apply to the DLQ called ABC1.DEAD.LETTER.QUEUE, owned by the queue manager ABC1.QUEUE.MANAGER.

If you do not specify the DLQ or the queue manager as shown above, the default queue manager for the installation is used along with the DLQ belonging to that queue manager.

The runmqdlq command takes its input from stdin; you associate the rules table with runmqdlq by redirecting stdin from the rules table.

To run the DLQ handler you must be authorized to access both the DLQ itself and any message queues to which messages on the DLQ are forwarded. For the DLQ handler to put messages on queues with the authority of the user ID in the message context, you must also be authorized to assume the identity of other users.

For more information about the runmqdlq command, see runmqdlq.

»The sample DLQ handler, amqsdlq«

Parent topic: »The WebSphere MQ dead-letter queue handler«

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14000_

6.5.1.1. The sample DLQ handler, amqsdlq

In addition to the DLQ handler invoked using the runmqdlq command, WebSphere® MQprovides the source of a sample DLQ handler, amqsdlq, whose function is similar to that provided by runmqdlq. You can customize amqsdlq to provide a DLQ handler that meets your requirements. For example, you might decide that you want a DLQ handler that can process messages without dead-letter headers. (Both the default DLQ handler and the sample, amqsdlq, process only those messages on the DLQ that begin with a dead-letter header, MQDLH. Messages that do not begin with an MQDLH are identified as being in error, and remain on the DLQ indefinitely.)

In WebSphere MQ for Windows, the source of amqsdlq is supplied in the directory:

```
c:\Program Files\IBM\WebSphere MQ\tools\c\samples\dlq
```

and the compiled version is supplied in the directory:

```
c:\Program Files\IBM\WebSphere MQ\tools\c\samples\bin
```

In WebSphere MQ for UNIX systems, the source of amqsdlq is supplied in the directory:

/opt/mqm/samp/dlq (/usr/mqm/samp/dlq on AIX®)

and the compiled version is supplied in the directory:

/opt/mqm/samp/bin (/usr/mqm/samp/bin on AIX)

Parent topic: »Invoking the DLQ handler«

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14010_

6.5.2. The DLQ handler rules table

The DLQ handler rules table defines how the DLQ handler is to process messages that arrive on the DLQ. There are two types of entry in a rules table:

- The first entry in the table, which is optional, contains *control data*.
- All other entries in the table are *rules* for the DLQ handler to follow. Each rule consists of a *pattern* (a set of message characteristics) that a message is matched against, and an *action* to be taken when a message on the DLQ matches the specified pattern. There must be at least one rule in a rules table.

Each entry in the rules table comprises one or more keywords.

»Control data«

»Rules (patterns and actions)«

»Rules table conventions«
The syntax, structure and contents of the DLQ handler rules table must adhere to these

conventions.

Parent topic: »The WebSphere MQ dead-letter queue handler«

This build: January 26, 2011 11:55:08

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14020_

6.5.2.1. Control data

This section describes the keywords that you can include in a control-data entry in a DLQ handler rules table. Note the following:

- The default value for a keyword, if any, is underlined.
- The vertical line (|) separates alternatives, only one of which can be specified.
- All keywords are optional.

INPUTQ (*QueueName*|' ')

The name of the DLQ you want to process:

1. Any INPUTQ value you supply as a parameter to the runmqdlq command overrides any INPUTQ value in the rules table.
2. If you do not specify an INPUTQ value as a parameter to the runmqdlq command, but you do specify a value in the rules table, the INPUTQ value in the rules table is used.
3. If no DLQ is specified or you specify INPUTQ(' ') in the rules table, the name of the DLQ belonging to the queue manager whose name is supplied as a parameter to the runmqdlq command is used.
4. If you do not specify an INPUTQ value as a parameter to the runmqdlq command or as a value in the rules table, the DLQ belonging to the queue manager named on the INPUTQM keyword in the rules table is used.

INPUTQM (*QueueManagerName*|' ')

The name of the queue manager that owns the DLQ named on the INPUTQ keyword:

1. Any INPUTQM value you supply as a parameter to the runmqdlq command overrides any INPUTQM value in the rules table.
2. If you do not specify an INPUTQM value as a parameter to the runmqdlq command, the INPUTQM value in the rules table is used.
3. If no queue manager is specified or you specify INPUTQM(' ') in the rules table, the default queue manager for the installation is used.

RETRYINT (*Interval*|60)

The interval, in seconds, at which the DLQ handler should reprocess messages on the DLQ that could not be processed at the first attempt, and for which repeated attempts have been requested. By default, the retry interval is 60 seconds.

WAIT (YES|NO|*nnn*)

Whether the DLQ handler should wait for further messages to arrive on the DLQ when it detects that there are no further messages that it can process.

YES

The DLQ handler waits indefinitely.

NO

The DLQ handler ends when it detects that the DLQ is either empty or contains no messages that it can process.

nnn

The DLQ handler waits for *nnn* seconds for new work to arrive before ending, after it detects that the queue is either empty or contains no messages that it can process.

Specify WAIT (YES) for busy DLQs, and WAIT (NO) or WAIT (*nnn*) for DLQs that have a low level of activity. If the DLQ handler is allowed to terminate, invoke it again using triggering. For more information about triggering, see the WebSphere MQ Application Programming Guide.

An alternative to including control data in the rules table is to supply the names of the DLQ and its queue manager as input parameters to the runmqdlq command. If you specify a value both in the rules table and as input to the runmqdlq command, the value specified on the runmqdlq command takes precedence.

If you include a control-data entry in the rules table, it must be the first entry in the table.

Parent topic: The DLQ handler rules table

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14030_

6.5.2.2. Rules (patterns and actions)

Here is an example rule from a DLQ handler rules table:

```
PERSIST(MQPER_PERSISTENT) REASON (MQRC_PUT_INHIBITED) +
 ACTION (RETRY) RETRY (3)
```

This rule instructs the DLQ handler to make three attempts to deliver to its destination queue any persistent message that was put on the DLQ because MQPUT and MQPUT1 were inhibited.

All keywords that you can use on a rule are described in the rest of this section. Note the following:

- The default value for a keyword, if any, is underlined. For most keywords, the default value is * (asterisk), which matches any value.
- The vertical line (|) separates alternatives, only one of which can be specified.
- All keywords except ACTION are optional.

This section begins with a description of the pattern-matching keywords (those against which messages on the DLQ are matched), and then describes the action keywords (those that determine how the DLQ handler is to process a matching message).

The pattern-matching keywords

The action keywords

Parent topic: The DLQ handler rules table

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14040_

6.5.2.2.1. The pattern-matching keywords

The pattern-matching keywords, which you use to specify values against which messages on the DLQ are matched, are described below. All pattern-matching keywords are optional.

APPLIDAT (*ApplIdentityData*|*)

The *ApplIdentityData* value specified in the message descriptor, MQMD, of the message on the DLQ.

APPLNAME (*PutApplName*|*)

The name of the application that issued the MQPUT or MQPUT1 call, as specified in the *PutApplName* field of the message descriptor MQMD of the message on the DLQ.

APPLTYPE (*PutApplType*|*)

The *PutApplType* value, specified in the message descriptor MQMD, of the message on the DLQ.

DESTQ (*QueueName*|*)

The name of the message queue for which the message is destined.

DESTQM (*QueueManagerName*|*)

The name of the queue manager of the message queue for which the message is destined.

FEEDBACK (*Feedback*|*)

When the *MsgType* value is MQFB_REPORT, *Feedback* describes the nature of the report.

You can use symbolic names. For example, you can use the symbolic name MQFB_COA to identify those messages on the DLQ that need confirmation of their arrival on their destination queues.

FORMAT (*Format*|*)

The name that the sender of the message uses to describe the format of the message data.

MSGTYPE (*MsgType*|*)

The message type of the message on the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQMT_REQUEST to identify those messages on the DLQ that need replies.

PERSIST (*Persistence*|*)

The persistence value of the message. (The persistence of a message determines whether it survives restarts of the queue manager.)

You can use symbolic names. For example, you can use the symbolic name MQPER_PERSISTENT to identify messages on the DLQ that are persistent.

REASON (*ReasonCode*|*)

The reason code that describes why the message was put to the DLQ.

You can use symbolic names. For example, you can use the symbolic name MQRC_Q_FULL to identify those messages placed on the DLQ because their destination queues were full.

REPLYQ (*QueueName*|*)

The name of the reply-to queue specified in the message descriptor, MQMD, of the message on the DLQ.

REPLYQM (*QueueManagerName*|*)

The name of the queue manager of the reply-to queue, as specified in the message descriptor, MQMD, of the message on the DLQ.

USERID (*UserIdentifier*|*)

The user ID of the user who originated the message on the DLQ, as specified in the message descriptor, MQMD.

Parent topic: »Rules (patterns and actions)«

This build: January 26, 2011 11:55:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14050_

6.5.2.2.2. The action keywords

The action keywords, used to describe how a matching message is to be processed, are described below.

ACTION (DISCARD|IGNORE|RETRY|FWD)

The action to be taken for any message on the DLQ that matches the pattern defined in this rule.

DISCARD

Delete the message from the DLQ.

IGNORE

Leave the message on the DLQ.

RETRY

If the first attempt to put the message on its destination queue fails, try again. The RETRY keyword sets the number of tries made to implement an action. The RETRYINT keyword of the control data controls the interval between attempts.

FWD

Forward the message to the queue named on the FWDQ keyword.

You must specify the ACTION keyword.

FWDQ (*QueueName*|&DESTQ|&REPLYQ)

The name of the message queue to which to forward the message when ACTION (FWD) is

requested.

QueueName

The name of a message queue. FWDQ(' ') is not valid.

&DESTQ

Take the queue name from the *DestQName* field in the MQDLH structure.

&REPLYQ

Take the queue name from the *ReplyToQ* field in the message descriptor, MQMD.

To avoid error messages when a rule specifying FWDQ (&REPLYQ) matches a message with a blank *ReplyToQ* field, specify REPLYQ (?*) in the message pattern.

FWDQM (*QueueManagerName*|&DESTQM|&REPLYQM|' ')

The queue manager of the queue to which to forward a message.

QueueManagerName

The name of the queue manager of the queue to which to forward a message when ACTION (FWD) is requested.

&DESTQM

Take the queue manager name from the *DestQMgrName* field in the MQDLH structure.

&REPLYQM

Take the queue manager name from the *ReplyToQMgr* field in the message descriptor, MQMD.

' '

FWDQM(' '), which is the default value, identifies the local queue manager.

HEADER (YES|NO)

Whether the MQDLH should remain on a message for which ACTION (FWD) is requested. By default, the MQDLH remains on the message. The HEADER keyword is not valid for actions other than FWD.

PUTAUT (DEF|CTX)

The authority with which messages should be put by the DLQ handler:

DEF

Put messages with the authority of the DLQ handler itself.

CTX

Put the messages with the authority of the user ID in the message context. If you specify PUTAUT (CTX), you must be authorized to assume the identity of other users.

RETRY (*RetryCount*|1)

The number of times, in the range 1-999 999 999, to try an action (at the interval specified on the RETRYINT keyword of the control data). The count of attempts made by the DLQ handler to implement any particular rule is specific to the current instance of the DLQ handler; the count does not persist across restarts. If the DLQ handler is restarted, the count of attempts made to apply a rule is reset to zero.

Parent topic: »Rules (patterns and actions)«

This build: January 26, 2011 11:55:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14060_

6.5.2.3. Rules table conventions

The syntax, structure and contents of the DLQ handler rules table must adhere to these conventions.

The rules table must adhere to the following conventions:

- A rules table must contain at least one rule.
- Keywords can occur in any order.
- A keyword can be included only once in any rule.
- Keywords are not case-sensitive.
- A keyword and its parameter value must be separated from other keywords by at least one blank or comma.
- There can be any number of blanks at the beginning or end of a rule, and between

keywords, punctuation, and values.

- Each rule must begin on a new line.
- On Windows systems, the last rule in the table must end with a carriage return/line feed character. You can achieve this by ensuring that you press the Enter key at the end of the rule, so that the last line of the table is a blank line.
- For reasons of portability, the significant length of a line must not be greater than 72 characters.
- Use the plus sign (+) as the last nonblank character on a line to indicate that the rule continues from the first nonblank character in the next line. Use the minus sign (-) as the last nonblank character on a line to indicate that the rule continues from the start of the next line. Continuation characters can occur within keywords and parameters. For example:

```
APPLNAME('ABC+
  D')
```

  results in 'ABCD', and

```
APPLNAME('ABC-
  D')
```

  results in 'ABC D'.

- Comment lines, which begin with an asterisk (*), can occur anywhere in the rules table.
- Blank lines are ignored.
- Each entry in the DLQ handler rules table comprises one or more keywords and their associated parameters. The parameters must follow these syntax rules:
  - Each parameter value must include at least one significant character. The delimiting quotation marks in quoted values are not considered significant. For example, these parameters are valid:

```
FORMAT('ABC')        3 significant characters
FORMAT(ABC)          3 significant characters
FORMAT('A')          1 significant character
FORMAT(A)            1 significant character
FORMAT(' ')          1 significant character
```

    These parameters are invalid because they contain no significant characters:

```
FORMAT('')
FORMAT( )
FORMAT()
FORMAT
```

  - Wildcard characters are supported. You can use the question mark (?) instead of any single character, except a trailing blank; you can use the asterisk (*) instead of zero or more adjacent characters. The asterisk (*) and the question mark (?) are *always* interpreted as wildcard characters in parameter values.
  - Wildcard characters cannot be included in the parameters of these keywords: ACTION, HEADER, RETRY, FWDQ, FWDQM, and PUTAUT.
  - Trailing blanks in parameter values, and in the corresponding fields in the message on the DLQ, are not significant when performing wildcard matches. However, leading and embedded blanks within strings in quotation marks are significant to wildcard matches.
  - Numeric parameters cannot include the question mark (?) wildcard character. You can use the asterisk (*) instead of an entire numeric parameter, but not as part of a numeric parameter. For example, these are valid numeric parameters:

```
MSGTYPE(2)           Only reply messages are eligible
MSGTYPE(*)           Any message type is eligible
MSGTYPE('*')         Any message type is eligible
```

    However, MSGTYPE('2*') is not valid, because it includes an asterisk (*) as part of a numeric parameter.

  - Numeric parameters must be in the range 0-999 999 999. If the parameter value is in this range, it is accepted, even if it is not currently valid in the field to which the keyword relates. You can use symbolic names for numeric parameters.
  - If a string value is shorter than the field in the MQDLH or MQMD to which the keyword relates, the value is padded with blanks to the length of the field. If the value, excluding asterisks, is longer than the field, an error is diagnosed. For example,

these are all valid string values for an 8 character field:

```
'ABCDEFGH'            8 characters
'A*C*E*G*I'           5 characters excluding asterisks
'*A*C*E*G*I*K*M*O*'   8 characters excluding asterisks
```

- Enclose strings that contain blanks, lowercase characters, or special characters other than period (.), forward slash (/), underscore (_), and percent sign (%) in single quotation marks. Lowercase characters not enclosed in quotation marks are folded to uppercase. If the string includes a quotation, use two single quotation marks to denote both the beginning and the end of the quotation. When the length of the string is calculated, each occurrence of double quotation marks is counted as a single character.

Parent topic: »The DLQ handler rules table«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14070_

6.5.3. How the rules table is processed

The DLQ handler searches the rules table for a rule whose pattern matches a message on the DLQ. The search begins with the first rule in the table, and continues sequentially through the table. When the DLQ handler finds a rule with a matching pattern, it takes the action from that rule. The DLQ handler increments the retry count for a rule by 1 whenever it applies that rule. If the first try fails, the DLQ handler tries again until the number of tries matches the number specified on the RETRY keyword. If all attempts fail, the DLQ handler searches for the next matching rule in the table.

This process is repeated for subsequent matching rules until an action is successful. When each matching rule has been attempted the number of times specified on its RETRY keyword, and all attempts have failed, ACTION (IGNORE) is assumed. ACTION (IGNORE) is also assumed if no matching rule is found.

Note:

1. Matching rule patterns are sought only for messages on the DLQ that begin with an MQDLH. Messages that do not begin with an MQDLH are reported periodically as being in error, and remain on the DLQ indefinitely.
2. All pattern keywords can be allowed to default, such that a rule can consist of an action only. Note, however, that action-only rules are applied to all messages on the queue that have MQDLHs and that have not already been processed in accordance with other rules in the table.
3. The rules table is validated when the DLQ handler starts, and errors are flagged at that time. (Error messages issued by the DLQ handler are described in WebSphere MQ Messages.) You can make changes to the rules table at any time, but those changes do not come into effect until the DLQ handler restarts.
4. The DLQ handler does not alter the content of messages, the MQDLH, or the message descriptor. The DLQ handler always puts messages to other queues with the message option MQPMO_PASS_ALL_CONTEXT.
5. Consecutive syntax errors in the rules table might not be recognized because the rules table is designed to eliminate the generation of repetitive errors during validation.
6. The DLQ handler opens the DLQ with the MQOO_INPUT_AS_Q_DEF option.
7. Multiple instances of the DLQ handler can run concurrently against the same queue, using the same rules table. However, it is more usual for there to be a one-to-one relationship between a DLQ and a DLQ handler.

»Ensuring that all DLQ messages are processed«

Parent topic: »The WebSphere MQ dead-letter queue handler«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa14080_

6.5.3.1. Ensuring that all DLQ messages are processed

The DLQ handler keeps a record of all messages on the DLQ that have been seen but not removed. If you use the DLQ handler as a filter to extract a small subset of the messages from the DLQ, the DLQ handler still has to keep a record of those messages on the DLQ that it did not process. Also, the DLQ handler cannot guarantee that new messages arriving on the DLQ are seen, even if the DLQ is defined as first-in-first-out (FIFO). If the queue is not empty, the DLQ is periodically re-scanned to check all messages.

For these reasons, try to ensure that the DLQ contains as few messages as possible; if messages that cannot be discarded or forwarded to other queues (for whatever reason) are allowed to accumulate on the queue, the workload of the DLQ handler increases and the DLQ itself can fill up.

You can take specific measures to enable the DLQ handler to empty the DLQ. For example, try not to use ACTION (IGNORE), which leaves messages on the DLQ. (Remember that ACTION (IGNORE) is assumed for messages that are not explicitly addressed by other rules in the table.) Instead, for those messages that you would otherwise ignore, use an action that moves the messages to another queue. For example:

```
ACTION (FWD) FWDQ (IGNORED.DEAD.QUEUE) HEADER (YES)
```

Similarly, make the final rule in the table a catchall to process messages that have not been addressed by earlier rules in the table. For example, the final rule in the table could be something like this:

```
ACTION (FWD) FWDQ (REALLY.DEAD.QUEUE) HEADER (YES)
```

This forwards messages that fall through to the final rule in the table to the queue REALLY.DEAD.QUEUE, where they can be processed manually. If you do not have such a rule, messages are likely to remain on the DLQ indefinitely.

Parent topic: »How the rules table is processed«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14090_

6.5.4. An example DLQ handler rules table

The following example rules table contains a single control-data entry and several rules:

```
************************************************************************
*         An example rules table for the runmqdlq command              *
************************************************************************
* Control data entry
* ------------------
* If no queue manager name is supplied as an explicit parameter to
* runmqdlq, use the default queue manager for the machine.
* If no queue name is supplied as an explicit parameter to runmqdlq,
* use the DLQ defined for the local queue manager.
*
inputqm(' ')  inputq(' ')

* Rules
* -----
* We include rules with ACTION (RETRY) first to try to
* deliver the message to the intended destination.
* If a message is placed on the DLQ because its destination
* queue is full, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).

REASON(MQRC_Q_FULL) ACTION(RETRY) RETRY(5)

* If a message is placed on the DLQ because of a put inhibited
* condition, attempt to forward the message to its
* destination queue. Make 5 attempts at approximately
* 60-second intervals (the default value for RETRYINT).
```

```
REASON(MQRC_PUT_INHIBITED) ACTION(RETRY) RETRY(5)

* The AAAA corporation are always sending messages with incorrect
* addresses. When we find a request from the AAAA corporation,
* we return it to the DLQ (DEADQ) of the reply-to queue manager
* (&REPLYQM).
* The AAAA DLQ handler attempts to redirect the message.

MSGTYPE(MQMT_REQUEST) REPLYQM(AAAA.*) +
  ACTION(FWD) FWDQ(DEADQ) FWDQM(&REPLYQM)

* The BBBB corporation never do things by half measures. If
* the queue manager BBBB.1 is unavailable, try to
* send the message to BBBB.2

DESTQM(bbbb.1) +
  action(fwd) fwdq(&DESTQ) fwdqm(bbbb.2) header(no)

* The CCCC corporation considers itself very security
* conscious, and believes that none of its messages
* will ever end up on one of our DLQs.
* Whenever we see a message from a CCCC queue manager on our
* DLQ, we send it to a special destination in the CCCC organization
* where the problem is investigated.

REPLYQM(CCCC.*) +
  ACTION(FWD) FWDQ(ALARM) FWDQM(CCCC.SYSTEM)

* Messages that are not persistent run the risk of being
* lost when a queue manager terminates. If an application
* is sending nonpersistent messages, it should be able
* to cope with the message being lost, so we can afford to
* discard the message.   PERSIST(MQPER_NOT_PERSISTENT) ACTION(DISCARD)
* For performance and efficiency reasons, we like to keep
* the number of messages on the DLQ small.
* If we receive a message that has not been processed by
* an earlier rule in the table, we assume that it
* requires manual intervention to resolve the problem.
* Some problems are best solved at the node where the
* problem was detected, and others are best solved where
* the message originated. We don't have the message origin,
* but we can use the REPLYQM to identify a node that has
* some interest in this message.
* Attempt to put the message onto a manual intervention
* queue at the appropriate node. If this fails,
* put the message on the manual intervention queue at
* this node.

REPLYQM('?*') +
  ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION) FWDQM(&REPLYQM)

ACTION(FWD) FWDQ(DEADQ.MANUAL.INTERVENTION)
```

Parent topic: »The WebSphere MQ dead-letter queue handler«

This build: January 26, 2011 11:55:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14100_

6.6. Availability, recovery and restart

»Make your applications highly available by maintaining queue availability if a queue manager fails, and recover messages after server or storage failure.«

»Improve client application availability by using client reconnection to switch a client automatically between a group of queue managers, or to the new active instance of a multi-instance queue manager after a queue manager failure. Automatic client reconnect is not supported by WebSphere® MQ classes for Java. «

Improve server application availability on z/OS® by using queue sharing groups.

On Windows, UNIX, and IBM® i platforms deploy server applications to a multi-instance queue manager, which is configured to run as a single queue manager on multiple servers; if the

server running the active instance fails, execution is automatically switched to a standby instance of the same queue manager on a different server. If you configure server applications to run as queue manager services, they are restarted when a standby instance becomes the actively running queue manager instance.

You can configure WebSphere MQ as part of platform-specific clustering solution such as Microsoft Cluster Server, HA clusters on IBM i, or HA-CMP on AIX® and other UNIX clustering solutions.

Another way to increase server application availability is to deploy server applications to multiple computers in a queue manager cluster.

A messaging system ensures that messages entered into the system are delivered to their destination. WebSphere MQ can trace the route of a message as it moves from one queue manager to another using the dspmqrte command. If a system fails, messages can be recovered in various ways depending on the type of failure, and the way a system is configured.

WebSphere MQ ensures that messages are not lost by maintaining recovery logs of the activities of the queue managers that handle the receipt, transmission, and delivery of messages. It uses these logs for three types of recovery:

1. *Restart recovery*, when you stop WebSphere MQ in a planned way.
2. *Crash recovery*, when a failure stops WebSphere MQ.
3. *Media recovery*, to restore damaged objects.

In all cases, the recovery restores the queue manager to the state it was in when the queue manager stopped, except that any in-flight transactions are rolled back, removing from the queues any updates that were in-flight at the time the queue manager stopped. Recovery restores all persistent messages; nonpersistent messages might be lost during the process.

»Automatic client reconnection«
You can make your client applications reconnect automatically, without writing any additional code, by configuring a number of components.

»Stopping and removing queue managers manually«
If the standard methods for stopping and removing queue managers fail, try the methods described here.

»Using WebSphere MQ with high availability configurations«
If you want to operate your WebSphere MQ queue managers in a high availability (HA) configuration, you can set up your queue managers to work either with a high availability manager, such as PowerHA™ for AIX (formerly HACMP™) or the Microsoft Cluster Service (MSCS), or with WebSphere MQ multi-instance queue managers.

»Making sure that messages are not lost (logging)«
WebSphere MQ logs all the information you need to recover from a queue manager failure.

»Backing up and restoring WebSphere MQ«
Backing up queue managers and queue manager data.

»Recovery scenarios«
Types of failures and how to recover from them.

Parent topic: »Configuring and managing WebSphere MQ«

Related information
What is a queue-sharing group?

This build: January 26, 2011 11:55:12

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14470_

6.6.1. Automatic client reconnection

You can make your client applications reconnect automatically, without writing any additional code, by configuring a number of components.

Automatic client reconnection is *inline*. The connection is automatically restored at any

point in the client application program, and the handles to open objects are all restored.

In contrast, manual reconnection requires the client application to re-create a connection using MQCONN or MQCONNX, and to reopen objects. Automatic client reconnection is suitable for many, but not all client applications.

Table 1 lists the earliest release of WebSphere® MQ client support that must be installed on a client workstation. You must upgrade client workstations to one of these levels for an application to use automatic client reconnection. Table 2 lists other requirements to enable automatic client reconnection.

With program access to reconnection options, a client application can set reconnection options. Except for JMS and XMS clients, if a client application has access to reconnection options, it can also create an event handler to handle reconnection events.

An existing client application might be able to benefit from reconnection support, without recompilation and linking:

- For a non-JMS client, set the mqclient.ini environment variable DefRecon to set reconnection options. Use a CCDT to connect to a queue manager. If the client is to connect to a multi-instance queue manager, provide the network addresses of the active and standby queue manager instances in the CCDT.
- For a JMS client, set the reconnection options in the connection factory configuration. JMS clients running in a managed Java EE container are always reconnectable. The reconnection is managed by the Java EE resource adapter, rather than by the WebSphere MQ client support.

Table 1. Supported clients

Client interface	Client	Program access to reconnection options	Reconnection support
Messaging APIs	C, C++, COBOL, Unmanaged Visual Basic, XMS (Unmanaged XMS on Windows)	7.0.1	7.0.1
	JMS (JSE, and Java EE client container and managed containers)	7.0.1.3	7.0.1.3
	WebSphere MQ classes for Java	Not supported	Not supported
	Managed XMS and managed .NET clients: C#, Visual Basic,	Not supported	Not supported
Other APIs	Windows Communication Foundation (Unmanaged[1])	Not supported	7.0.1
	Windows Communication Foundation (Managed[1])	Not supported	Not supported
	Axis 1	Not supported	Not supported
	Axis 2	Not supported	7.0.1.3
	HTTP (web 2.0)	Not supported	7.0.1.3

1. Set managed or unmanaged mode in the WCF binding configuration.

Automatic reconnection has the following configuration requirements:

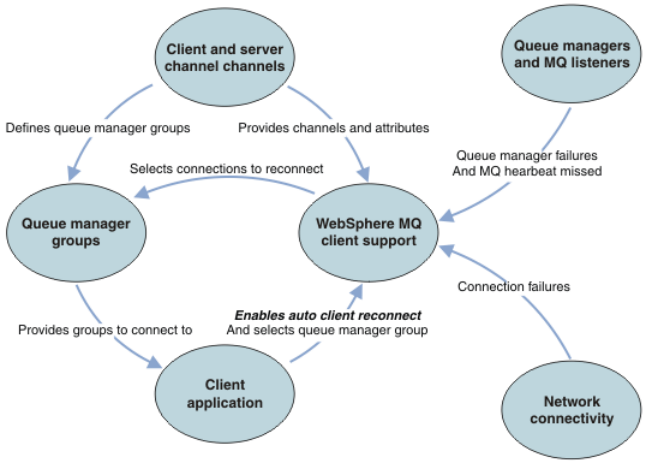
Table 2. Automatic reconnection configuration requirements

Component	Requirement	Effect of not meeting requirement
WebSphere MQ Client installation	See Table 1	MQRC_OPTIONS_ERROR
WebSphere MQ Server installation	Level 7.0.1	MQRC_OPTIONS_ERROR
Channel	SHARECNV > 0	MQRC_ENVIRONMENT_ERROR
Application environment	Must be threaded	MQRC_ENVIRONMENT_ERROR
MQI	One of:	MQCC_FAILED when a connection

	• MQCONNX with MQCNO Options set to MQCNO_RECONNECT or MQCNO_RECONNECT_Q_MGR. • Defrecon=YES|QMGR in mqclient.ini • In JMS set the CLIENTRECONNECTOPTIONS property of the connection factory.	is broken or queue manager ends or fails.

Figure 1 shows the main interactions between components that are involved in client reconnection.

Figure 1. Automatic client reconnection



Client application

The client application is a WebSphere MQ client.

- By default clients are not automatically reconnected. Enable the automatic client reconnection by setting the MQCONNX MQCNO Option MQCNO_RECONNECT or MQCNO_RECONNECT_Q_MGR.
- Many applications are written in such a way that they are able to take advantage of auto-reconnection with no additional coding. Enable automatic reconnection for existing programs, without making any coding changes, by setting the DefRecon attribute in the channels stanza of the mqclient.ini configuration file.
- Use one of these three options:
  1. Modify the program so that the logic is unaffected by reconnection. For example, you might have to issue MQI calls within the sync point, and resubmit backed-out transactions.
  2. Add an event handler to detect reconnection, and restore the state of the client application when the connection is reestablished.
  3. Do not enable auto-reconnection: instead, disconnect the client and issue a new MQCONN or MQCONNX MQI call to find another queue manager instance that is running in the same queue manager group.

  For further details about these three options, see Application recovery.
- Reconnecting to a queue manager of the same name does not guarantee that you have reconnected to the same instance of a queue manager.
  Use an MQCNO option MQCNO_RECONNECT_Q_MGR, to reconnect to an instance of the same queue manager.
- A client can register an event handler so that it can be informed the state of reconnection. The MQHCONN passed in the event handler cannot be used. The following reason codes are provided:

  MQRC_RECONNECTING

  The connection failed, and the system is attempting to reconnect. You receive multiple MQRC_RECONNECTING events if multiple reconnect attempts are made.

  MQRC_RECONNECTED

  The reconnection made and all handles successfully reestablished.

  MQRC_RECONNECT_FAILED

  The reconnection was not successful.

MQRC_RECONNECT_QMID_MISMATCH

A reconnectable connection specified MQCNO_RECONNECT_Q_MGR and the connection attempted to reconnect to a different queue manager.

MQRC_RECONNECT_Q_MGR_REQD

An option, such MQMO_MATCH_MSG_TOKEN in an MQGET call, was specified in the client program that requires reconnection to the same queue manager.

- A reconnectable client is able to reconnect automatically only *after* connecting. That is, the MQCONNX call itself is not tried again if it fails. For example, if you receive the return code 2543 - MQRC_STANDBY_Q_MGR from MQCONNX, reissue the call after a short delay.

MQRC_RECONNECT_INCOMPATIBLE

This reason code is returned when the application tries to use MQPMO_LOGICAL_ORDER (with MQPUT and MQPUT1) or MQGMO_LOGICAL_ORDER (with MQGET) when reconnect options are set. The reason for returning the reason code is to make sure that applications never use reconnect in such cases.

MQRC_CALL_INTERRUPTED

This reason code is returned when the connection breaks during the execution of Commit call and the client reconnects. An MQPUT of a persistent message outside the sync point also results in the same reason code being returned to the application.

Multi-instance queue managers

Simplify restarting WebSphere MQ client applications, after a multi-instance queue manager has activated its standby instance, by using automatic client reconnection.

The standby instance of a multi-instance queue manager is typically at a different network address to the active instance. Include the network addresses of both the instances in the client connection definition table (CCDT). Either provide a list of network addresses for the CONNAME parameter, or define multiple rows for the queue manager in the CCDT.

Commonly, WebSphere MQ clients reconnect to any queue manager in a queue manager group. Sometimes you want a WebSphere MQ client to reconnect only to the same queue manager. It might have an affinity to a queue manager. You can prevent a client from reconnecting to a different queue manager. Set the MQCNO option, MQCNO_RECONNECT_Q_MGR. The WebSphere MQ client fails if it reconnects to a different queue manager. If you set the MQCNO option, MQCNO_RECONNECT_Q_MGR, do not include other queue managers in the same queue manager group. The client returns an error if the queue manager it reconnects to is not the same queue manager as the one it connected to.

Queue manager groups

You can select whether the client application always connects and reconnects to a queue manager of the same name, to the same queue manager, or to any of a set of queue managers that are defined with the same QMNAME value in the client connection table.

- The queue manager *name* attribute, QMNAME, in the client channel definition is the name of a queue manager group.
- In your client application, if you set the value of the MQCONN or MQCONNX QmgrName parameter to a queue manager name, the client connects only to queue managers with that name. If you prefix the queue manager name with an asterisk(*), the client connects to any queue manager in the queue manager group with the same QMNAME value. For a full explanation, see Queue manager groups.

Queue sharing groups

Automatic client reconnection to z/OS® queue sharing groups uses the same mechanisms for reconnection as any other environment. The client will reconnect to the same selection of queue managers as is configured for the original connection. For example, when using the client channel definition table the administrator should ensure that all entries in the table resolve to the same z/OS queue sharing group.

Client and server channel definitions

Client and server channel definitions define the groups of queue managers a client application can reconnect to. The definitions govern the selection and timing of reconnections, and other factors, such as security; see the related topics. The most relevant channel attributes to consider for reconnection are listed in two groups:

Client connection attributes

AFFINITY

Connection affinity.

CLNTWGHT

Client channel weight.

CONNAME

Connection information.

HBINT

Heartbeat interval. Set the heartbeat interval on the server connection channel.

KAINT

Keepalive interval. Set the keepalive interval on the server connection channel.

QMNAME

Queue manager name.

Server connection attributes

HBINT

Heartbeat interval. Set the heartbeat interval on the client connection channel.

KAINT

Keepalive interval. Set the keepalive interval on the client connection channel.

KAINT is a network layer heartbeat, and HBINT is a WebSphere MQ heartbeat between the client and the queue manager. Setting these heartbeats to a shorter time serves two purposes:

1. By simulating activity on the connection, network layer software that is responsible for closing inactive connections is less likely to shut down your connection.
2. If the connection is shut down, the delay before the broken connection is detected, is shortened.

The default TCP/IP keepalive interval is two hours. Consider setting the KAINT and HBINT attributes to a shorter time. Do not assume that the normal behavior of a network suits the needs of automatic reconnection. For example, some firewalls can shut down an inactive TCP/IP connection after as little as 10 minutes.

Network connectivity

Only network failures that are passed to the WebSphere MQ client by the network, are handled by the automatic reconnection capability of the client.

- Reconnections performed automatically by the transport are invisible to WebSphere MQ.
- Setting HBINT helps to deal with network failures that are invisible to WebSphere MQ.

Queue managers and WebSphere MQ listeners

Client reconnection is triggered by server failure, queue manager failure, network connectivity failure, and by an administrator switching over to another queue manager instance.

- If you are using a multi-instance queue manager, an additional cause of client reconnection occurs when you switch control from the active queue manager instance to a standby instance.
- Ending a queue manager using the default endmqm command, does not trigger automatic client reconnection. Add the -r option on the endmqm command to request automatic client reconnection, or the -s option to transfer to a standby queue manager instance after shutting down.

WebSphere MQ client automatic reconnection support

If you use the automatic client reconnection support in the WebSphere MQ client, the client application automatically reconnects and continues processing without you issuing an MQCONN or MQCONNX MQI call to reconnect to the queue manager.

- Automatic client reconnection is triggered by one of the following occurrences:
  - queue manager failure
  - ending a queue manager and specifying the -r, reconnect, option on the endmqm command
- The MQCONNX MQCNO options control whether you have enabled the automatic client reconnection. The options are described in Reconnection options.

- Automatic client reconnection issues MQI calls on behalf of your application to restore the connection handle and the handles to other open objects, so that your program can resume normal processing after it has processed any MQI errors that resulted from the broken connection. See Recovery of an automatically reconnected client.
- If you have written a channel exit program for the connection, the exit receives these additional MQI calls.
- You can register a reconnection event handler, which is triggered when reconnection begins and when it finishes.

Although reconnection takes no more than a minute, reconnection can take longer because a queue manager might have numerous resources to manage. During this time, a client application might be holding locks on that do not belong to WebSphere MQ resources. There is a timeout value you can configure to limit the time a client waits for reconnection. The value (in seconds) is set in the mqclient.ini file.

```
Channels:
    MQReconnectTimeout = 1800
```

No reconnection attempts are made after the timeout has expired. When the system detects that the timeout has expired it returns a MQ_RECONNECT_FAILED error.

Parent topic: Availability, recovery and restart

This build: January 26, 2011 11:56:21

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70190_

6.6.2. Stopping and removing queue managers manually

If the standard methods for stopping and removing queue managers fail, try the methods described here.

Stopping a queue manager manually

Removing queue managers manually

Parent topic: Availability, recovery and restart

This build: January 26, 2011 11:56:08

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22280_

6.6.2.1. Stopping a queue manager manually

The standard way of stopping queue managers is by using the endmqm command. To stop a queue manager manually, use one of the procedures described below. For details of how to perform operations on queue managers using control commands, see Managing queue managers.

Stopping queue managers in WebSphere MQ for Windows
Stopping queue managers in WebSphere® MQ for Windows.

Stopping queue managers in WebSphere MQ for UNIX systems
Stopping queue managers in WebSphere MQ for UNIX.

Parent topic: Stopping and removing queue managers manually

This build: January 26, 2011 11:56:08

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22290_

6.6.2.1.1. Stopping queue managers in WebSphere MQ for Windows

Stopping queue managers in WebSphere® MQ for Windows.

To stop a queue manager running under WebSphere MQ for Windows:

1. **List the names (IDs) of the processes that are running, by using the Windows Task Manager.**
2. **End the processes by using Windows Task Manager, or the taskkill command, in the following order (if they are running):**

AMQZMUC0	Critical process manager
AMQZXMA0	Execution controller
AMQZFUMA	OAM process
AMQZLAA0	LQM agents
AMQZLSA0	LQM agents
AMQZMUF0	Utility Manager
AMQZMGR0	Process controller
AMQZMUR0	Restartable process manager
AMQFQPUB	Publish Subscribe process
AMQFCXBA	Broker worker process
AMQRMPPA	Process pooling process
»AMQCRSTA«	Non-threaded responder job process
»AMQCRS6B«	LU62 receiver channel and client connection
AMQRRMFA	The repository process (for clusters)
AMQZDMAA	Deferred message processor
AMQPCSEA	The command server
»RUNMQTRM«	Invoke a trigger monitor for a server
»RUNMQDLQ«	Invoke dead-letter queue handler
RUNMQCHI	The channel initiator process
RUNMQLSR	The channel listener process
AMQXSSVN	Shared memory servers
AMQZTRCN	Trace

3. **Stop the WebSphere MQ service from Administration tools > Services on the Windows Control Panel.**
4. **If you have tried all methods and the queue manager has not stopped, reboot your system.**

The Windows Task Manager and the tasklist command give limited information about tasks. For more information to help to determine which processes relate to a particular queue manager, consider using a tool such as *Process Explorer* (procexp.exe), available for download from the Microsoft Web site (www.microsoft.com).

Parent topic: »Stopping a queue manager manually«

This build: January 26, 2011 11:56:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22300_

6.6.2.1.2. Stopping queue managers in WebSphere MQ for UNIX systems

Stopping queue managers in WebSphere® MQ for UNIX.

To stop a queue manager running under WebSphere MQ for UNIX systems:

1. **Find the process IDs of the queue manager programs that are still running by using the ps command. For example, if the queue manager is called QMNAME, use the following command:**

   ```
   ps -ef | grep QMNAME
   ```

2. **End any queue manager processes that are still running. Use the kill command, specifying the process IDs discovered by using the ps command.**
   End the processes in the following order:

amqzmuc0	Critical process manager

amqzxma0	Execution controller
amqzfuma	OAM process
amqzlaa0	LQM agents
amqzlsa0	LQM agents
amqzmuf0	Utility Manager
amqzmur0	Restartable process manager
amqzmgr0	Process controller
amqfqpub	Publish Subscribe process
amqfcxba	Broker worker process
amqrmppa	Process pooling process
amqcrsta	Non-threaded responder job process
amqcrs6b	LU62 receiver channel and client connection
amqrrmfa	The repository process (for clusters)
amqzdmaa	Deferred message processor
amqpcsea	The command server
runmqtrm	Invoke a trigger monitor for a server
runmqdlq	Invoke dead-letter queue handler
runmqchi	The channel initiator process
runmqlsr	The channel listener process
amqxssvn	Shared memory servers
amqztrcn	Trace

Note: You can use the kill -9 command to end processes that fail to stop.

If you stop the queue manager manually, FFSTs might be taken, and FDC files placed in /var/mqm/errors. Do not regard this as a defect in the queue manager.

The queue manager will restart normally, even after you have stopped it using this method.

Parent topic: Stopping a queue manager manually

This build: January 26, 2011 11:56:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22310_

6.6.2.2. Removing queue managers manually

If you want to delete the queue manager after stopping it manually, use the dltmqm command. If, for some reason, this command fails to delete the queue manager, use the manual processes described here.

Removing queue managers in WebSphere MQ for Windows

Removing queue managers in WebSphere MQ for UNIX systems

Parent topic: Stopping and removing queue managers manually

This build: January 26, 2011 11:56:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22320_

6.6.2.2.1. Removing queue managers in WebSphere MQ for Windows

If you encounter problems with the dltmqm command in WebSphere® MQ for Windows, use the following procedure to delete a queue manager:

1. **Type REGEDIT from the command prompt to start the Registry Editor.**
2. **Select the HKEY_LOCAL_MACHINE window.**

3. Navigate the tree structure in the left-hand pane of the Registry Editor to the following key:
   HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion
   Make a note of the values within this key called WorkPath and LogPath. Within each of the directories named by these values, you are going to delete a subdirectory containing the data for the queue manager that you are trying to delete. You now need to find out the name of the subdirectory which corresponds to your queue manager.
4. Navigate the tree structure to the following key:
   HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\ Configuration\QueueManager
   Within this key there is a key for each of the queue managers on this computer containing the configuration information for the queue manager. The name of this queue manager key is the name of the subdirectory in which the queue manager's data is stored in the file system. By default, this name is the same as the queue manager name, but the name might be a transformation of the queue manager name.
5. Examine the keys within the current key. Look for the key that contains a value called Name. Name contains the name of the queue manager you are trying to delete. Make a note of the name of the key containing the name of the queue manager you are trying to delete. This is the subdirectory name.
6. Locate the queue manager data directory. The name of this directory is the WorkPath followed by the subdirectory name. Delete this directory, and all subdirectories and files.
7. Locate the queue manager's log directory. The name of this directory is the LogPath followed by the subdirectory name. Delete this directory, and all subdirectories and files.
8. Remove the registry entries that refer to the deleted queue manager. First, navigate the tree structure in the Registry Editor to the following key:
   HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\ Configuration\DefaultQueueManager
9. If the value called Name within this key matches the name of the queue manager you are deleting, delete the DefaultQueueManager key.
10. Navigate the tree to the following key:
    HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\ Configuration\Services
11. Within this key, delete the key whose name matches the subdirectory name of the queue manager which you are deleting.
12. Navigate the tree to the following key:
    HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\ Configuration\QueueManager
13. Within this key, delete the key whose name matches the subdirectory name of the queue manager which you are deleting.

»Removing queue managers from the automatic startup list«

Parent topic: »Removing queue managers manually«

This build: January 26, 2011 11:56:08

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22330_

6.6.2.2.1.1. Removing queue managers from the automatic startup list

If for any reason the WebSphere® MQ Explorer cannot be used to change the startup state of a particular queue manager, use the following routine to carry out the same procedure manually:

1. Stop the WebSphere MQ Explorer either from the taskbar icon or from the control panel.
2. Type REGEDIT on the command line.
3. Select the HKEY_LOCAL_MACHINE window.
4. Navigate the tree structure to find the following key:
   LOCAL_MACHINE\Software\IBM\MQSeries\CurrentVersion\Configuration\
   Services\<QMgrName>\QueueManager
5. Change the startup value to 0. (1 means automatic and 0 means manual.)
6. Close the Registry Editor.
7. Run amqmdain regsec.

Parent topic: »Removing queue managers in WebSphere MQ for Windows«

This build: January 26, 2011 11:56:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22340_

6.6.2.2.2. Removing queue managers in WebSphere® MQ for UNIX systems

The manual removal of a queue manager is potentially very disruptive, particularly if multiple queue managers are being used on a single system. This is because, to completely remove a queue manager, you must delete files, shared memory, and semaphores.

If you need to delete a queue manager manually, use the following procedure:

1. Stop the queue manager running, and execute the following command, as user mqm:

   ```
   amqiclen -x -m QMGR
   ```

   This ensures that all IPC resources that are specifically reserved for queue manager *QMGR* are removed.
2. Locate the queue manager directory from the configuration file /var/mqm/mqs.ini. To do this, look for the QueueManager stanza naming the queue manager to be deleted. Its Prefix and Directory attributes identify the queue manager directory. For a Prefix attribute of <Prefix> and a Directory attribute of <Directory>, the full path to the queue manager directory is: <Prefix>/qmgrs/<Directory>
3. Locate the queue manager log directory from the qm.ini configuration file in the queue manager directory. The LogPath attribute of the Log stanza identifies this directory.
4. Delete the queue manager directory, all subdirectories and files.
5. Delete the queue manager log directory, all subdirectories and files.
6. Remove the queue manager's QueueManager stanza from the /var/mqm/mqs.ini configuration file.
7. If the queue manager being deleted is also the default queue manager, remove the DefaultQueueManager stanza from the /var/mqm/mqs.ini configuration file.
8. »Delete the queue manager sockets directory, /var/mqm/sockets/<Directory>, and all sub directories and files. «

Parent topic: »Removing queue managers manually«

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22350_

6.6.3. Using WebSphere MQ with high availability configurations

If you want to operate your WebSphere® MQ queue managers in a high availability (HA) configuration, you can set up your queue managers to work either with a high availability manager, such as PowerHA™ for AIX® (formerly HACMP™) or the Microsoft Cluster Service (MSCS), or with WebSphere MQ multi-instance queue managers.

You need to be aware of the following configuration definitions:

Queue manager clusters

Groups of two or more queue managers on one or more computers, providing automatic interconnection, and allowing queues to be shared among them for load balancing and redundancy.

HA clusters

Groups of two or more computers, and resources such as disks and networks, connected together and configured in such a way that, if one fails, a high availability manager, such as HACMP (UNIX) or MSCS (Windows) performs a *failover*. The failover transfers the state data of applications from the failing computer to another computer in the cluster and re-initiates their operation there. This provides high availability of services running within the HA cluster. The relationship between WebSphere MQ clusters and HA clusters is described in Relationship of HA clusters to queue manager clusters.

Multi-instance queue managers

Instances of the same queue manager configured on two or more computers. By starting

multiple instances, one instance becomes the active instance and the other instances become standbys. If the active instance fails, a standby instance running on a different computer automatically takes over. You can use multi-instance queue managers to configure your own highly available messaging systems based on WebSphere MQ, without requiring a cluster technology such as HACMP or MSCS. HA clusters and multi-instance queue managers are alternative ways of making queue managers highly available. Do not combine them by putting a multi-instance queue manager in an HA cluster.

Differences between multi-instance queue managers and HA clusters

Multi-instance queue managers and HA clusters are alternative ways to achieve high availability for your queue managers. Here are some points that highlight the differences between the two approaches.

Multi-instance queue managers include the following features:

- Basic failover support integrated into WebSphere MQ
- Faster failover than HA cluster
- Simple configuration and operation
- Integration with WebSphere MQ Explorer

Limitations of multi-instance queue managers include:

- Highly available, high performance networked storage required
- More complex network configuration because queue manager changes IP address when it fails over
- A standby instance must be restarted manually when a failover occurs

HA clusters include the following features:

- The ability to coordinate multiple resources, such as an application server or database
- More flexible configuration options including clusters comprising more than two nodes
- Can failover multiple times without operator intervention
- Takeover of queue manager's IP address as part of the failover

Limitations of HA clusters include:

- Additional product purchase and skills are required
- Disks which can be switched between the nodes of the cluster are required
- Configuration of HA clusters is relatively complex
- Failover is rather slow historically, but recent HA cluster products are improving this
- Unnecessary failovers can occur if there are shortcomings in the scripts that are used to monitor resources such as queue managers

Relationship of HA clusters to queue manager clusters

Queue manager clusters reduce administration and provide load balancing of messages across instances of queue manager cluster queues. They also offer higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still access surviving instances of a queue manager cluster queue. However, queue manager clusters alone do not provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. HA clusters provide these features. The two types of cluster can be used together to good effect.

»Using WebSphere MQ with a high availability cluster on UNIX«
You can use WebSphere MQ with a high availability (HA) cluster on UNIX platforms: for example, PowerHA for AIX (formerly HACMP), Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite.

»Supporting the Microsoft Cluster Service (MSCS)«
Introducing and setting up MSCS to support failover of virtual servers.

»Multi-instance queue managers«
A »multi-instance« queue manager restarts automatically on a standby server.

Parent topic: »Availability, recovery and restart«

This build: January 26, 2011 11:56:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70240_

6.6.3.1. Using WebSphere MQ with a high availability cluster on UNIX

You can use WebSphere® MQ with a high availability (HA) cluster on UNIX platforms: for example, PowerHA™ for AIX® (formerly HACMP™), Veritas Cluster Server, HP Serviceguard, or a Red Hat Enterprise Linux cluster with Red Hat Cluster Suite.

Before WebSphere MQ Version 7.0.1, SupportPac MC91 was provided to assist in configuring HA clusters. WebSphere MQ Version 7.0.1 provided a greater degree of control than previous versions over where queue managers store their data. This makes it easier to configure queue managers in an HA cluster. Most of the scripts provided with SupportPac MC91 are no longer required, and the SupportPac is withdrawn.

This section introduces HA cluster configurations, the relationship of HA clusters to queue manager clusters, WebSphere MQ clients, and WebSphere MQ operating in an HA cluster, and guides you through the steps and provides example scripts that you can adapt to configure queue managers with an HA cluster.

Refer to the HA cluster documentation particular to your environment for assistance with the configuration steps described in this section.

HA cluster configurations

In this section the term *node* is used to refer to the entity that is running an operating system and the HA software; "computer", "system" or "machine" or "partition" or "blade" might be considered synonyms in this usage. You can use WebSphere MQ to help set up either standby or takeover configurations, including mutual takeover where all cluster nodes are running WebSphere MQ workload.

A *standby* configuration is the most basic HA cluster configuration in which one node performs work while the other node acts only as standby. The standby node does not perform work and is referred to as idle; this configuration is sometimes called *cold standby*. Such a configuration requires a high degree of hardware redundancy. To economize on hardware, it is possible to extend this configuration to have multiple worker nodes with a single standby node. The point of this is that the standby node can take over the work of any other worker node. This configuration is still referred to as a standby configuration and sometimes as an "N+1" configuration.

A *takeover* configuration is a more advanced configuration in which all nodes perform some work and critical work can be taken over in the event of a node failure.

A *one-sided takeover* configuration is one in which a standby node performs some additional, noncritical and unmovable work. This configuration is similar to a standby configuration but with (noncritical) work being performed by the standby node.

A *mutual takeover* configuration is one in which all nodes are performing highly available (movable) work. This type of HA cluster configuration is also sometimes referred to as "Active/Active" to indicate that all nodes are actively processing critical workload.

With the extended standby configuration or either of the takeover configurations it is important to consider the peak load that might be placed on a node that can take over the work of other nodes. Such a node must possess sufficient capacity to maintain an acceptable level of performance.

Relationship of HA clusters to queue manager clusters

Queue manager clusters reduce administration and provide load balancing of messages across instances of queue manager cluster queues. They also offer higher availability than a single queue manager because, following a failure of a queue manager, messaging applications can still access surviving instances of a queue manager cluster queue. However, queue manager clusters alone do not provide automatic detection of queue manager failure and automatic triggering of queue manager restart or failover. HA clusters provide these features. The two types of cluster can be used together to good effect.

WebSphere MQ clients

WebSphere MQ clients that are communicating with a queue manager that might be subject to a restart or takeover must be written to tolerate a broken connection and must repeatedly attempt to reconnect. WebSphere MQ Version 7 introduced features in the processing of the Client Channel Definition Table that assist with connection availability and workload

balancing; however these are not directly relevant when working with a failover system.

The Extended Transactional Client (ETC), which allows a WebSphere MQ Client to participate in two-phase transactions, must always connect to the same queue manager. The ETC cannot use techniques such as an IP load balancer to select from a list of queue managers. When you use an HA product, a queue manager maintains its identity (name and address) whichever node it is running on, so the ETC can be used with queue managers that are under HA control.

WebSphere MQ operating in an HA cluster

All HA clusters have the concept of a unit of failover. This is a set of definitions that contains all the resources that make up the highly available service. The unit of failover includes the service itself and all other resources upon which it depends.

HA solutions use different terms for a unit of failover:

- On PowerHA for AIX the unit of failover is called a *resource group*.
- On Veritas Cluster Server it is known as a *service group*.
- On Serviceguard it is called a *package*.

This topic uses the term *resource group* to mean a unit of failover.

The smallest unit of failover for WebSphere MQ is a queue manager. Typically, the resource group containing the queue manager also contains shared disks in a volume group or disk group that is reserved exclusively for use by the resource group, and the IP address that is used to connect to the queue manager. It is also possible to include other WebSphere MQ resources, such as a listener or a trigger monitor in the same resource group, either as separate resources, or under the control of the queue manager itself.

A queue manager that is to be used in an HA cluster must have its data and logs on disks that are shared between the nodes in the cluster. The HA cluster ensures that only one node in the cluster at a time can write to the disks. The HA cluster can use a monitor script to monitor the state of the queue manager.

It is possible to use a single shared disk for both the data and logs that are related to the queue manager. However, it is normal practice to use separate shared file systems so that they can be independently sized and tuned.

Figure 1. HA cluster

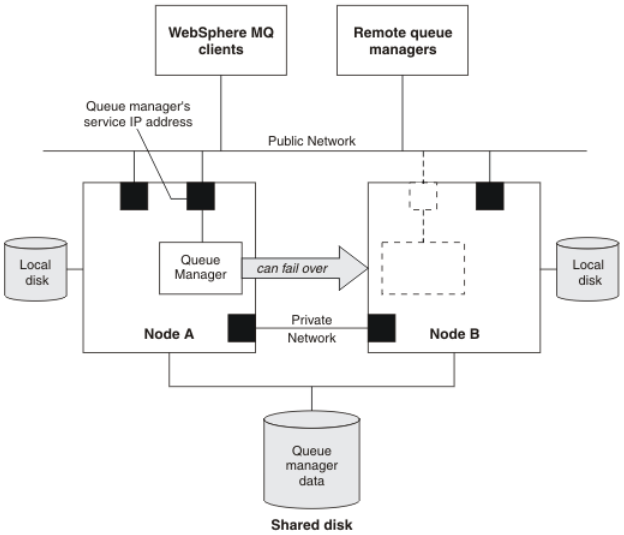


Figure 1 illustrates a HA cluster with two nodes. The HA cluster is managing the availability of a queue manager which has been defined in a resource group. This is an active/passive or cold standby configuration, because only one node, node A, is currently running a queue manager. The queue manager was created with its data and log files on a shared disk. The queue manager has a service IP address which is also managed by the HA cluster. The queue manager depends on the shared disk and its service IP address. When the HA cluster fails the queue manager over from node A to node B, it first moves the queue manager's dependent resources onto node B and then starts the queue manager.

If the HA cluster contains more than one queue manager, your HA cluster configuration might result in two or more queue managers running on the same node after a failover. Each queue manager in the HA cluster must be assigned its own port number, which it uses on whichever cluster node it happens to be active at any particular time.

Generally, the HA cluster runs as the root user. WebSphere MQ runs as the mqm user. Administration of WebSphere MQ is granted to members of the mqm group. Ensure that the mqm user and group both exist on all HA cluster nodes. The user ID and group ID must be

consistent across the cluster. Administration of WebSphere MQ by the root user is not allowed; scripts that start, stop, or monitor scripts must switch to the mqm user.

Note: WebSphere MQ must be installed correctly on all nodes; you cannot share the product executable files.

»Configuring the shared disks«
A WebSphere MQ queue manager in an HA cluster requires data files and log files to be in common named remote file systems on a shared disk.

»Creating a queue manager for use in a high availability (HA) cluster«
The first step towards using a queue manager in a high availability cluster is to create the queue manager on one of the nodes.

»Adding queue manager configuration information to other nodes in a high availability (HA) cluster«
You must add the queue manager configuration to the other nodes in the HA cluster.

»Starting a queue manager under control of a high availability (HA) cluster«
The queue manager is represented in the HA cluster as a resource. The HA cluster must be able to start and stop the queue manager. In most cases you can use a shell script to start the queue manager. You must make these scripts available at the same location on all nodes in the cluster, either using a network filesystem or by copying them to each of the local disks.

»Stopping a queue manager under the control of a high availability (HA) cluster«
In most cases, you can use a shell script to stop a queue manager. Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to stop the queue manager under control of your HA cluster.

»Monitoring a queue manager«
It is usual to provide a way for the high availability (HA) cluster to monitor the state of the queue manager periodically. In most cases, you can use a shell script for this. Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to make additional monitoring checks specific to your environment.

»Putting the queue manager under control of the high availability (HA) cluster«
You must configure the queue manager, under control of the HA cluster, with the queue manager's IP address and shared disks.

»Deleting a queue manager from a high availability (HA) cluster node«
You might want to remove a queue manager from a node that is no longer required to run the queue manager.

Parent topic: »Using WebSphere MQ with high availability configurations«

This build: January 26, 2011 11:56:22

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70230_

6.6.3.1.1. Configuring the shared disks

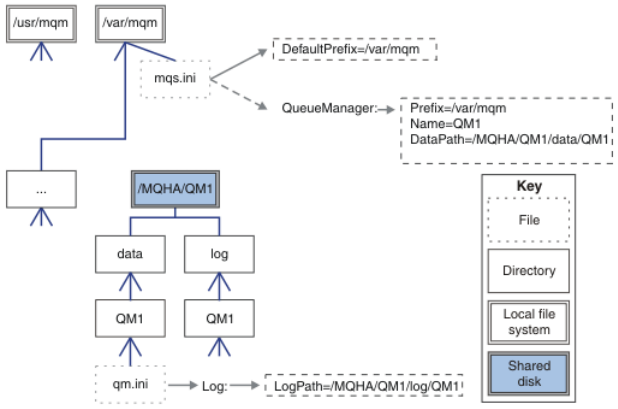
A WebSphere® MQ queue manager in an HA cluster requires data files and log files to be in common named remote file systems on a shared disk.

To configure the shared disks, complete the following steps:

1. Decide the names of the mount points for the queue manager's file systems. For example, /MQHA/qmgrname/data for the queue manager's data files and /MQHA/qmgrname/log for its log files.
2. Create a volume group (or disk group) to contain the queue manager's data and log files. This volume group is managed by the high availability (HA) cluster in the same resource group as the queue manager.
3. Create the file systems for the queue manager's data and log files in the volume group.
4. For each node in turn, create the mount points for the file systems and make sure that the file systems can be mounted. The mqm user must own the mount points.

Figure 1 shows a possible layout for a queue manager in an HA cluster. The queue manager's data and log directories are both on the shared disk which is mounted at /MQHA/QM1. This disk is switched between the nodes of the HA cluster when failover occurs so that the data is available wherever the queue manager is restarted. The mqs.ini file has a stanza for the QM1 queue manager. The Log stanza in the qm.ini file has a value for LogPath.

Figure 1. Shared named data and log directories



Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

Related concepts
Changing configuration information on UNIX systems

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70250_

6.6.3.1.2. Creating a queue manager for use in a high availability (HA) cluster

The first step towards using a queue manager in a high availability cluster is to create the queue manager on one of the nodes.

To create a queue manager for use in an HA cluster, select one of the nodes in the cluster on which to create the queue manager. On this node complete the following steps:

1. Mount the queue manager's file systems on the node.
2. Create the queue manager by using the crtmqm command. For example:
   crtmqm -md /MQHA/qmgrname/data -ld /MQHA/qmgrname/log qmgrname
3. Start the queue manager manually by using the strmqm command.
4. Complete any initial configuration of the queue manager, such as creating queues and channels, and setting the queue manager to start a listener automatically when the queue manager starts.
5. Stop the queue manager by using the endmqm command.
6. Use the dspmqinf command to display the addmqinf command that you can use in a later task, which is documented in Adding queue manager configuration information to other nodes in a high availability (HA) cluster:
   dspmqinf -o command qmgrname
   where qmgrname is the name of the queue manager.
7. The addmqinf command that is displayed will be similar to the following example:

   ```
   addmqinf -sQueueManager -vName=qmgrname -vDirectory=qmgrname \
            -vPrefix=/var/mqm -vDataPath=/MQHA/qmgrname/data/qmgrname
   ```

   Make a careful note of the displayed command.
8. Unmount the queue manager's file systems.

You are now ready to complete the steps described in Adding queue manager configuration information to other nodes in a high availability (HA) cluster.

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

Related reference
crtmqm
dspmqinf
addmqinf
strmqm
endmqm

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70260_

6.6.3.1.3. Adding queue manager configuration information to other nodes in a high availability (HA) cluster

You must add the queue manager configuration to the other nodes in the HA cluster.

Before you complete this task, you must have completed the steps in Creating a queue manager for use in a high availability (HA) cluster.

To add the configuration information for the queue manager to each of other nodes in the HA cluster, complete the following steps on each of the other nodes:

1. Mount the queue manager file systems.
2. Add the queue manager configuration information to the node, either by editing /var/mqm/mqs.ini directly, or by issuing the addmqinf command that was displayed by the dspmqinf command in steps 6 and 7 in Creating a queue manager for use in a high availability (HA) cluster.
3. Start and stop the queue manager to verify the configuration.
4. Unmount the queue manager file systems.

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

Related concepts
Changing configuration information on UNIX systems

Related reference
addmqinf

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70320_

6.6.3.1.4. Starting a queue manager under control of a high availability (HA) cluster

The queue manager is represented in the HA cluster as a resource. The HA cluster must be able to start and stop the queue manager. In most cases you can use a shell script to start the queue manager. You must make these scripts available at the same location on all nodes in the cluster, either using a network filesystem or by copying them to each of the local disks.

Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to start the queue manager under the control of your HA cluster.

The following shell script is an example of how to switch from the HA cluster user to the mqm user so that the queue manager can be successfully started:

```
#!/bin/ksh

# A simple wrapper script to switch to the mqm user.

su mqm -c name_of_your_script $*
```

The following shell script is an example of how to start a queue manager without making any

assumptions about the current state of the queue manager. Note that it uses an extremely abrupt method of ending any processes that belong to the queue manager:

```
#!/bin/ksh
#
# This script robustly starts the queue manager.
#
# The script must be run by the mqm user.

# The only argument is the queue manager name. Save it as QM variable
QM=$1

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi

# End any queue manager processes which might be running.

srchstr="( |-m)$QM *.*$"
for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
                amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
                amqzfuma amqzdmaa amqzmuf0 amqzmur0 amqzmgr0
 do
  ps -ef | tr "\t" " " | grep $process | grep -v grep | \
    egrep "$srchstr" | awk '{print $2}'| \
       xargs kill -9 > /dev/null 2>&1
done

# It is now safe to start the queue manager.
# The strmqm command does not use the -x flag.
strmqm ${QM}
```

You can modify the script to start other related programs.

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70280_

6.6.3.1.5. Stopping a queue manager under the control of a high availability (HA) cluster

In most cases, you can use a shell script to stop a queue manager. Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to stop the queue manager under control of your HA cluster.

The following script is an example of how to immediately stop without making assumptions about the current state of the queue manager. The script must be run by the mqm user; it might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm (an example shell script is provided in Starting a queue manager under control of a high availability (HA) cluster):

```
#!/bin/ksh
#
# The script ends the QM by using two phases, initially trying an immediate
# end with a time-out and escalating to a forced stop of remaining
# processes.
#
# The script must be run by the mqm user.
#
# There are two arguments: the queue manager name and a timeout value.
QM=$1
TIMEOUT=$2

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi
```

```
if [ -z "$TIMEOUT" ]
then
  echo "ERROR! No timeout specified"
  exit 1
fi

for severity in immediate brutal
do
  # End the queue manager in the background to avoid
  # it blocking indefinitely. Run the TIMEOUT timer
  # at the same time to interrupt the attempt, and try a
  # more forceful version. If the brutal version fails,
  # nothing more can be done here.

  echo "Attempting ${severity} end of queue manager '${QM}'"
  case $severity in

  immediate)
    # Minimum severity of endmqm is immediate which severs connections.
    # HA cluster should not be delayed by clients
    endmqm -i ${QM} &
    ;;

  brutal)
    # This is a forced means of stopping queue manager processes.

    srchstr="( |-m)$QM *.*$"
    for process in amqzmuc0 amqzxma0 amqfcxba amqfqpub amqpcsea amqzlaa0 \
                amqzlsa0 runmqchi runmqlsr amqcrsta amqrrmfa amqrmppa \
                amqzfuma amqzdmaa amqzmuf0 amqzmur0 amqzmgr0
    do
      ps -ef | tr "\t" " " | grep $process | grep -v grep | \
          egrep "$srchstr" | awk '{print $2}'| \
              xargs kill -9 > /dev/null 2>&1
    done

  esac

  TIMED_OUT=yes
  SECONDS=0
  while (( $SECONDS < ${TIMEOUT} ))
  do
   TIMED_OUT=yes
   i=0
   while [ $i -lt 5 ]
   do
     # Check for execution controller termination
     srchstr="( |-m)$QM *.*$"
     cnt=`ps -ef | tr "\t" " " | grep amqzxma0 | grep -v grep | \
       egrep "$srchstr" | awk '{print $2}' | wc -l `
     i=`expr $i + 1`
     sleep 1
     if [ $cnt -eq 0 ]
     then
       TIMED_OUT=no
       break
     fi
   done

   if [ ${TIMED_OUT} = "no" ]
   then
     break
   fi

   echo "Waiting for ${severity} end of queue manager '${QM}'"
   sleep 1
  done # timeout loop

  if [ ${TIMED_OUT} = "yes" ]
  then
    continue         # to next level of urgency
  else
    break             # queue manager is ended, job is done
  fi
```

```
    done # next phase
```

Parent topic: Using WebSphere MQ with a high availability cluster on UNIX

This build: January 26, 2011 11:56:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70290_

6.6.3.1.6. Monitoring a queue manager

It is usual to provide a way for the high availability (HA) cluster to monitor the state of the queue manager periodically. In most cases, you can use a shell script for this. Examples of suitable shell scripts are given here. You can tailor these to your needs and use them to make additional monitoring checks specific to your environment.

Note: PowerHA™ for AIX® does not provide a way of supplying a parameter to the monitoring program for the queue manager. You must create a separate monitoring program for each queue manager, that encapsulates the queue manager name. Here is an example of a script used on AIX to encapsulate the queue manager name:

```
#!/bin/ksh
su mqm -c name_of_monitoring_script qmname
```

The following script is not robust to the possibility that runmqsc hangs. Typically, HA clusters treat a hanging monitoring script as a failure and are themselves robust to this possibility.

The script does, however, tolerate the queue manager being in the starting state. This is because it is common for the HA cluster to start monitoring the queue manager as soon as it has started it. Some HA clusters distinguish between a starting phase and a running phase for resources, but it is necessary to configure the duration of the starting phase. Because the time taken to start a queue manager depends on the amount of work that it has to do, it is very hard to choose a maximum time that starting a queue manager takes. If you choose a value that is too low, the HA cluster incorrectly assumes that the queue manager has failed when it simply has not yet completed starting up. This could result in an endless sequence of failovers.

This script must be run by the mqm user; it might therefore be necessary to wrap this script in a shell script to switch the user from the HA cluster user to mqm (an example shell script is provided in Starting a queue manager under control of a high availability (HA) cluster):

```
#!/bin/ksh
#
# This script tests the operation of the queue manager.
#
# An exit code is generated by the runmqsc command:
# 0  => Either the queue manager is starting or the queue manager is running and responds
#       Either is OK.
# >0 => The queue manager is not responding and not starting.
#
# This script must be run by the mqm user.
QM=$1

if [ -z "$QM" ]
then
  echo "ERROR! No queue manager name supplied"
  exit 1
fi

# Test the operation of the queue manager. Result is 0 on success, non-zero on error.
echo "ping qmgr" | runmqsc ${QM} > /dev/null 2>&1
pingresult=$?

if [ $pingresult -eq 0 ]
then # ping succeeded

  echo "Queue manager '${QM}' is responsive"
  result=0

else # ping failed
```

```
    # Don't condemn the queue manager immediately, it might be starting.
    srchstr="( |-m)$QM *.*$"
    cnt=`ps -ef | tr "\t" " " | grep strmqm | grep "$srchstr" | grep -v grep \
                    | awk '{print $2}' | wc -l`
    if [ $cnt -gt 0 ]
    then
      # It appears that the queue manager is still starting up, tolerate
      echo "Queue manager '${QM}' is starting"
      result=0
    else
      # There is no sign of the queue manager starting
      echo "Queue manager '${QM}' is not responsive"
      result=$pingresult
    fi

  fi

  exit $result
```

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70300_

6.6.3.1.7. Putting the queue manager under control of the high availability (HA) cluster

You must configure the queue manager, under control of the HA cluster, with the queue manager's IP address and shared disks.

To define a resource group to contain the queue manager and all of its associated resources, complete the following steps:

1. Create the resource group containing the queue manager, the queue manager's volume or disk group, and the queue manager's IP address. The IP address is a virtual IP address, not the IP address of the computer.
2. Verify that the HA cluster correctly switches the resources between the cluster nodes and is ready to control the queue manager.

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70270_

6.6.3.1.8. Deleting a queue manager from a high availability (HA) cluster node

You might want to remove a queue manager from a node that is no longer required to run the queue manager.

To remove the queue manager from a node in an HA cluster, complete the following steps:

1. Remove the node from the HA cluster so that the HA cluster will no longer attempt to activate the queue manager on this node.
2. Use the following rmvmqinf command to remove the queue manager's configuration information:
   rmvmqinf *qmgrname*

To completely delete the queue manager, use the dltmqm command. However, be aware that this completely deletes the queue manager's data and log files. When you have deleted the queue manager, you can use the rmvmqinf command to remove remaining configuration information from the other nodes.

Parent topic: »Using WebSphere MQ with a high availability cluster on UNIX«

Related reference
dltmqm
rmvmqinf

This build: January 26, 2011 11:56:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70310_

6.6.3.2. Supporting the Microsoft Cluster Service (MSCS)

Introducing and setting up MSCS to support failover of virtual servers.

This information applies to WebSphere® MQ for Windows only.

The Microsoft Cluster Service (MSCS) enables you to connect servers into a *cluster*, giving higher availability of data and applications, and making it easier to manage the system. MSCS can automatically detect and recover from server or application failures.

MSCS supports *failover* of *virtual servers*, which correspond to applications, Web sites, print queues, or file shares (including their disk spindles, files, IP addresses, and so on).

Failover is the process by which MSCS detects a failure in an application on one computer in the cluster, and shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and re-initiates the application there.

This chapter introduces MSCS clusters and describes setting up MSCS support in the following sections:

- Introducing MSCS clusters
- Setting up WebSphere MQ for MSCS clustering

Then tells you how to configure WebSphere MQ for MSCS clustering, in the following sections:

- Creating a queue manager for use with MSCS
- Moving a queue manager to MSCS storage
- Putting a queue manager under MSCS control
- Removing a queue manager from MSCS control

And then gives some useful hints on using MSCS with WebSphere MQ, and details the WebSphere MQ MSCS support utility programs, in the following sections:

- Hints and tips on using MSCS
- WebSphere MQ MSCS support utility programs

Introducing MSCS clusters

Setting up WebSphere MQ for MSCS clustering

Creating a queue manager for use with MSCS

Moving a queue manager to MSCS storage

Putting a queue manager under MSCS control
The tasks involved in placing a queue manager under MSCS control, including prerequisite tasks.

Summary of looksAlive and isAlive polling

Removing a queue manager from MSCS control

Hints and tips on using MSCS

WebSphere MQ MSCS support utility programs

Parent topic: Using WebSphere MQ with high availability configurations

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14110_

6.6.3.2.1. Introducing MSCS clusters

»Before we start to look at MSCS clusters, we need to distinguish between them, WebSphere® MQ clusters and multi-instance queue managers:«

»Multi-instance queue managers«

»Multi-instance queue managers are instances of the same queue manager configured on two or more computers. By starting multiple instances, one instance becomes the active instance and the other instances become standbys. If the active instance fails, a standby instance running on a different computer automatically takes over. You can use multi-instance queue managers to configure your own highly available messaging systems based on WebSphere MQ, without requiring a cluster technology such as MSCS. You cannot combine multi-instance queue managers with MSCS clustering.«

WebSphere MQ clusters

WebSphere MQ clusters are groups of two or more queue managers on one or more computers, providing automatic interconnection, and allowing queues to be shared amongst them for load balancing and redundancy.

MSCS clusters

MSCS clusters are groups of two or more computers, connected together and configured in such a way that, if one fails, MSCS performs a *failover*, transferring the state data of applications from the failing computer to another computer in the cluster and re-initiating their operation there.

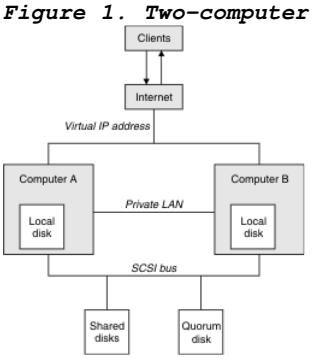
In the rest of this book, *clusters* means WebSphere MQ clusters. In this chapter, *clusters* always means MSCS clusters.

Let us start by looking at a two-machine cluster. A two-machine cluster comprises two computers (for example, A and B) that are jointly connected to a network for client access using a *virtual IP address*. They might also be connected to each other by one or more private networks. A and B share at least one disk for the server applications on each to use. There is also another shared disk, which must be a redundant array of independent disks (*RAID*) Level 1, for the exclusive use of MSCS; this is known as the *quorum* disk. MSCS monitors both computers to check that the hardware and software are running correctly.

In a simple setup such as this, both computers have all the applications installed on them, but only computer A runs with live applications; computer B is just running and waiting. If computer A encounters any one of a range of problems, MSCS shuts down the disrupted application in an orderly manner, transfers its state data to the other computer, and re-initiates the application there. This is known as a *failover*. Applications can be made *cluster-aware* so that they interact fully with MSCS and failover gracefully.

A typical setup for a two-computer cluster is as shown in Figure 1.

Figure 1. Two-computer MSCS cluster



Each computer can access the shared disk, but only one at a time, under the control of MSCS. In the event of a failover, MSCS switches the access to the other computer. The shared disk itself is usually a RAID, but need not be.

Each computer is connected to the external network for client access, and each has an IP address. However an external client, communicating with this cluster, is aware of only one *virtual IP address*, and MSCS routes the IP traffic within the cluster appropriately.

MSCS also performs its own communications between the two computers, either over one or more private connections or over the public network, in order to monitor their states using the

heartbeat, synchronize their databases, and so on.

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14120_

6.6.3.2.2. Setting up WebSphere MQ for MSCS clustering

You configure WebSphere® MQ for clustering by making the queue manager the unit of failover to MSCS. You define a queue manager as a resource to MSCS, which can then monitor it, and transfer it to another computer in the cluster if there is a problem.

To set your system up for this, you start by installing WebSphere MQ on each computer in the cluster. WebSphere MQ for Windows Quick Beginnings tells you how to do this.

The queue managers themselves need to exist only on the computer on which you create them. In the event of a failover, the MSCS initiates the queue managers on the other computer. The queue managers, however, must have their log and data files on a cluster shared disk, and not on a local drive. If you have a queue manager already installed on a local drive, you can migrate it using a tool provided with WebSphere MQ; see Moving a queue manager to MSCS storage. If you want to create new queue managers for use with MSCS, see Creating a queue manager for use with MSCS.

After installation and migration, use the MSCS Cluster Administrator to make MSCS aware of your queue managers; see Putting a queue manager under MSCS control.

If you decide to remove a queue manager from MSCS control, use the procedure described in Removing a queue manager from MSCS control.

- »Setup symmetry«
- »MSCS security«
- »Using multiple queue managers with MSCS«
- »Cluster modes«

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14130_

6.6.3.2.2.1. Setup symmetry

When an application switches from one node to the other it must behave in the same way, regardless of node. The best way of ensuring this is to make the environments identical. If you can, set up a cluster with identical hardware, operating system software, product software, and configuration on each computer. In particular, ensure that all the required software installed on the two computers is identical in terms of version, maintenance level, SupportPacs, paths and exits (as described WebSphere MQ for Windows Quick Beginnings), and that there is a common namespace (security environment) as described in MSCS security.

Parent topic: »Setting up WebSphere MQ for MSCS clustering«

This build: January 26, 2011 11:55:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14140_

6.6.3.2.2.2. MSCS security

Start by making sure you have identical software installations on each computer in the cluster, as described in WebSphere MQ for Windows Quick Beginnings.

For successful MSCS security, follow these guidelines:

- Create a common namespace (security environment) across the cluster.
- Make the nodes of the MSCS cluster members of a domain, within which the user account that is the *cluster owner* is a domain account.
- Make the other user accounts on the cluster also domain accounts, so that they are available on both nodes. This is automatically the case if you already have a domain, and the accounts relevant to WebSphere® MQ are domain accounts. If you do not currently have a domain, consider setting up a *mini-domain* to cater for the cluster nodes and relevant accounts. Your aim is to make your cluster of two computers look like a single computing resource.
  Remember that an account that is local to one computer does not exist on the other one. Even if you create an account with the same name on the other computer, its security identifier (SID) is different, so, when your application is moved to the other node, the permissions do not exist on that node.

During a failover or move, WebSphere MQ MSCS support ensures that all files that contain queue manager objects have equivalent permissions on the destination node. Explicitly, the code checks that the Administrators and mqm groups, and the SYSTEM account, have full control, and that if Everyone had read access on the old node, that permission is added on the destination node.

You can use a domain account to run your WebSphere MQ Service. Make sure that it exists in the local mqm group on each computer in the cluster.

Parent topic: »Setting up WebSphere MQ for MSCS clustering«

This build: January 26, 2011 11:55:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14150_

6.6.3.2.2.3. Using multiple queue managers with MSCS

If you are running more than one queue manager on a computer, you can choose one of the following setups:

- All the queue managers in a single group. In this configuration, if a problem occurs with any queue manager, all the queue managers in the group failover to the other computer as a group.
- A single queue manager in each group. In this configuration, if a problem occurs with the queue manager, it alone fails over to the other computer without affecting the other queue managers.
- A mixture of the first two setups.

Parent topic: »Setting up WebSphere MQ for MSCS clustering«

This build: January 26, 2011 11:55:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14160_

6.6.3.2.2.4. Cluster modes

There are two modes in which you might run a cluster system with WebSphere® MQ:

- Active/Passive
- Active/Active

Note: If you are using MSCS together with the Microsoft Transaction Server (COM+), you cannot use Active/Active mode.

»Active/Passive mode«

»Active/Active mode«

Parent topic: »Setting up WebSphere MQ for MSCS clustering«

This build: January 26, 2011 11:55:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14170_

6.6.3.2.2.4.1. Active/Passive mode

In Active/Passive mode, computer A has the running application on it, and computer B is backup, only being used when MSCS detects a problem.

You can use this mode with only one shared disk, but, if any application causes a failover, all the applications must be transferred as a group (because only one computer can access the shared disk at a time).

You can configure MSCS with A as the *preferred* computer. Then, when computer A has been repaired or replaced and is working properly again, MSCS detects this and automatically switches the application back to computer A.

If you run more than one queue manager, consider having a separate shared disk for each. Then put each queue manager in a separate group in MSCS. In this way, any queue manager can failover to the other computer without affecting the other queue managers.

Parent topic: »Cluster modes«

This build: January 26, 2011 11:55:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14180_

6.6.3.2.2.4.2. Active/Active mode

In Active/Active mode, computers A and B both have running applications, and the groups on each computer are set to use the other computer as backup. If a failure is detected on computer A, MSCS transfers the state data to computer B, and reinitiates the application there. computer B then runs its own application and A's.

For this setup you need at least two shared disks. You can configure MSCS with A as the preferred computer for A's applications, and B as the preferred computer for B's applications. After failover and repair, each application automatically ends up back on its own computer.

For WebSphere® MQ this means that you could, for example, run two queue managers, one on each of A and B, with each exploiting the full power of its own computer. After a failure on computer A, both queue managers would run on computer B. This would mean sharing the power of the one computer, with a reduced ability to process large quantities of data at speed. However, your critical applications would still be available while you find and repair the fault on A.

Parent topic: »Cluster modes«

This build: January 26, 2011 11:55:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14190_

6.6.3.2.3. Creating a queue manager for use with MSCS

This procedure ensures that a new queue manager is created in such a way that it is suitable for preparing and placing under MSCS control.

You start by creating the queue manager with all its resources on a local drive, and then migrate the log files and data files to a shared disk. (You can reverse this operation.) Do not attempt to create a queue manager with its resources on a shared drive.

You can create a queue manager for use with MSCS in two ways, either from a command prompt, or in the WebSphere® MQ Explorer. The advantage of doing using a command prompt is that the

queue manager is created *stopped* and set to *manual startup*, which is ready for MSCS. (The WebSphere MQ Explorer automatically starts a new queue manager and sets it to automatic startup after creation. You have to change this.)

- Creating a queue manager from a command prompt
- Creating a queue manager using the WebSphere MQ Explorer

Parent topic: Supporting the Microsoft Cluster Service (MSCS)

This build: January 26, 2011 11:55:10

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14200_

6.6.3.2.3.1. Creating a queue manager from a command prompt

1. Ensure that you have the environment variable MQSPREFIX set to refer to a local drive, for example C:\WebSphere MQ. If you change this, reboot the machine so that the System account picks up the change. If you do not set the variable, the queue manager is created in the WebSphere® MQ default directory for queue managers.
2. Create the queue manager using the crtmqm command. For example, to create a queue manager called mscs_test in the default directory, use:

   ```
   crtmqm mscs_test
   ```

3. Proceed to Moving a queue manager to MSCS storage.

Parent topic: Creating a queue manager for use with MSCS

This build: January 26, 2011 11:55:10

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14210_

6.6.3.2.3.2. Creating a queue manager using the WebSphere MQ Explorer

1. Start the WebSphere® MQ Explorer from the Start menu.
2. In the Navigator View, expand the tree nodes to find the Queue Managers tree node.
3. Right-click the Queue Managers tree node, and select New->Queue Manager. The Create Queue Manager panel is displayed.
4. Complete the dialog (Step 1), then click Next>.
5. Complete the dialog (Step 2), then click Next>.
6. Complete the dialog (Step 3), ensuring that Start Queue Manager and Create Server Connection Channel are not selected, then click Next>.
7. Complete the dialog (Step 4), then click Finish.
8. Proceed to Moving a queue manager to MSCS storage.

Parent topic: Creating a queue manager for use with MSCS

This build: January 26, 2011 11:55:10

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14220_

6.6.3.2.4. Moving a queue manager to MSCS storage

This procedure configures an existing queue manager to make it suitable for putting under MSCS control. To achieve this, you move the log files and data files to shared disks to make them available to the other computer in the event of a failure. For example, the existing queue manager might have paths such as C:\WebSphere MQ\log\<QMname> and C:\WebSphere

MQ\qmgrs\<QMname>. Do *not* try to move the files by hand; use the utility program supplied as part of WebSphere® MQ MSCS Support as described below.

If the queue manager being moved uses SSL connections and the SSL key repository is in the queue manager data directory on the local machine, then the key repository will be moved with the rest of the queue manager to the shared disk. By default, the queue manager attribute that specifies the SSL key repository location, SSLKEYR, is set to <mqmtop>\qmgrs\QMGRNAME\ssl\key, which is under the queue manager data directory. The hamvmqm command does not modify this queue manager attribute. In this situation you must modify the queue manager attribute, SSLKEYR, using the WebSphere MQ Explorer or the MQSC command ALTER QMGR, to point to the new SSL key repository file.

The procedure is:

1. Shut down the queue manager, and check that there are no errors.
2. If the queue manager's log files or queue files are already stored on a shared disk, skip the rest of this procedure and proceed directly to Putting a queue manager under MSCS control.
3. Make a full media backup of the queue files and log files and store the backup in a safe place (see Queue manager log files for why this is important).
4. If you already have a suitable shared disk resource proceed to step 6. Otherwise, using the MSCS Cluster Administrator to create a resource of type *shared disk* with sufficient capacity to store the queue manager log files and data (queue) files.
5. Test the shared disk by using the MSCS Cluster Administrator to move it from one cluster node to the other and back again.
6. Make sure that the shared disk is online on the cluster node where the queue manager log and data files are stored locally.
7. Run the utility program to move the queue manager as follows:

   hamvmqm /m *qmname* /dd "*e:\WebSphere MQ*" /ld "*e:\WebSphere MQ*\log"

   substituting your queue manager name for *qmname*, your shared disk drive letter for *e*, and your chosen directory for *WebSphere MQ*. The directories are created if they do not already exist.
8. Test the queue manager to ensure that it works, using the WebSphere MQ Explorer. For example:
   a. Right-click the queue manager tree node, then select Start. The queue manager starts.
   b. Right-click the Queues tree node, then select New->Local Queue..., and give the queue a name.
   c. Click Finish.
   d. Right-click the queue, then select Put Test Message.... The Put Test Message panel is displayed.
   e. Type some message text, then click Put Test Message, and close the panel.
   f. Right-click the queue, then select Browse Messages.... The Message Browser panel is displayed.
   g. Ensure your message is on the queue, then click Close . The Message Browser panel closes.
   h. Right-click the queue, then select Clear Messages.... The messages on the queue are cleared.
   i. Right-click the queue, then select Delete.... A confirmation panel is displayed, click OK. The queue is deleted.
   j. Right-click the queue manager tree node, then select Stop.... The End Queue Manager panel is displayed.
   k. Click OK. The queue manager stops.
9. As WebSphere MQ Administrator ensure that the startup attribute of the queue manager is set to manual. In the WebSphere MQ Explorer, set the Startup field to manual in the queue manager properties panel.
10. Proceed to Putting a queue manager under MSCS control.

Parent topic: Supporting the Microsoft Cluster Service (MSCS)

This build: January 26, 2011 11:55:10

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14230_

6.6.3.2.5. Putting a queue manager under MSCS control

The tasks involved in placing a queue manager under MSCS control, including prerequisite tasks.

Before you put a queue manager under MSCS control

Before you put a queue manager under MSCS control, perform the following tasks:

1. Ensure that WebSphere® MQ and its MSCS Support are installed on both machines in the cluster and that the software on each computer is identical, as described in Setting up WebSphere MQ for MSCS clustering.
2. Use the haregtyp utility program to register WebSphere MQ as an MSCS resource type on all the cluster nodes. See WebSphere MQ MSCS support utility programs for additional information.
3. If you have not yet created the queue manager, see Creating a queue manager for use with MSCS.
4. If you have created the queue manager, or it already exists, ensure that you have carried out the procedure in Moving a queue manager to MSCS storage.
5. Stop the queue manager, if it is running, using either a command prompt or the WebSphere MQ Explorer.
6. Test MSCS operation of the shared drives before going on to the procedure below.

Windows Server 2008

To place a queue manager under MSCS control on Windows Server 2008, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the Failover Cluster Management tool.
3. Right-click Failover Cluster Management > Manage a Cluster ... to open a connection to the cluster.
4. In contrast to the group scheme used in the MSCS Cluster Administrator on previous versions of Windows, the Failover Cluster Management tool uses the concept of services and applications. A configured service or application contains all the resources necessary for one application to be clustered. You can configure a queue manager under MSCS as follows:
   a. Right-click Services and Applications > Configure a Service or Application ... to start the configuration wizard.
   b. Select Other Server on the "Select Service or Application" panel.
   c. Select an appropriate IP address as a client access point.
      This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does not verify that the address can be reached.
   d. Assign a storage device for exclusive use by the queue manager. This device needs to be created as a resource instance before it can be assigned.
      You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.
      The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.
      For each drive resource, set possible owners to both nodes. Set dependent resources to none.
   e. Select the IBM MQSeries MSCS resource on the "Select Resource Type" panel.
   f. Complete the remaining steps in the wizard.
5. Before bringing the resource online, the IBM MQSeries MSCS resource needs additional configuration:
   a. Select the newly defined service which contains a resource called 'New IBM MQSeries MSCS'.

b. Right-click Properties on the MQ resource.

c. Configure the resource:

- Name; choose a name that makes it easy to identify which queue manager it is for.
- Run in a separate Resource Monitor; for better isolation
- Possible owners; set both nodes
- Dependencies; add the drive and IP address for this queue manager.
  Warning: Failure to add these dependencies means that WebSphere MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some WebSphere MQ processes could be blocked from running.
- Parameters; as follows:
  - QueueManagerName (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
  - PostOnlineCommand (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see PostOnlineCommand and PreOfflineCommand.
  - PreOfflineCommand (optional); you can specify a program to run whenever the queue manager resource changes its state from online to offline. For more details see PostOnlineCommand and PreOfflineCommand.

    Note: The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 60000 ms. These defaults can only be modified after the resource definition has been completed. For further details see Summary of looksAlive and isAlive polling.

d. Optionally, set a preferred node (but note the comments in Using preferred nodes)

e. The *Failover Policy* is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.

6. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the WebSphere MQ Explorer. For example:
   a. Right-click the Queues tree node, then select New->Local Queue..., and give the queue a name.
   b. Click Finish. The queue is created, and displayed in the content view.
   c. Right-click the queue, then select Put Test Message.... The Put Test Message panel is displayed.
   d. Type some message text, then click Put Test Message, and close the panel.
   e. Right-click the queue, then select Browse Messages.... The Message Browser panel is displayed.
   f. Ensure that your message is on the queue, then click Close . The Message Browser panel closes.
   g. Right-click the queue, then select Clear Messages.... The messages on the queue are cleared.
   h. Right-click the queue, then select Delete.... A confirmation panel is displayed, click OK. The queue is deleted.
7. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.
8. Simulate a failover.
   In the MSCS Cluster Administrator, right-click the group containing the queue manager and select Move Group. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in Moving a queue manager to MSCS storage.) You can also right-click and select Initiate Failure; the action (local restart or failover) depends on the current state and the configuration settings.

Windows 2003

To place a queue manager under MSCS control on Windows 2003, use the following procedure:

1. Log in to the cluster node computer hosting the queue manager, or log in to a remote workstation as a user with cluster administration permissions, and connect to the cluster node hosting the queue manager.
2. Start the MSCS Cluster Administrator.
3. Open a connection to the cluster.

4. Create an MSCS group to be used to contain the resources for the queue manager. Name the group in such a way that it is obvious which queue manager it relates to. Each group can contain multiple queue managers, as described in Using multiple queue managers with MSCS. Use the group for all the remaining steps.
5. Create a resource instance for each of the SCSI logical drives that the queue manager uses.
   You can use one drive to store both the logs and queue files, or you can split them up across drives. In either case, if each queue manager has its own shared disk, ensure that all drives used by this queue manager are exclusive to this queue manager, that is, that nothing else relies on the drives. Also ensure that you create a resource instance for every drive that the queue manager uses.
   The resource type for a drive depends on the SCSI support you are using; refer to your SCSI adapter instructions. There might already be groups and resources for each of the shared drives. If so, you do not need to create the resource instance for each drive. Move it from its current group to the one created for the queue manager.
   For each drive resource, set possible owners to both nodes. Set dependent resources to none.
6. Create a resource instance for the IP address.
   Create an IP address resource (resource type *IP address*). This address should be an unused IP address to be used by clients and other queue managers to connect to the *virtual* queue manager. This IP address is not the normal (static) address of either node; it is an additional address that *floats* between them. Although MSCS handles the routing of this address, it does not verify that the address can be reached.
7. Create a resource instance for the queue manager.
   Create a resource of type *IBM® WebSphere MQ MSCS*.The wizard prompts you for various items, including the following:
   - Name; choose a name that makes it easy to identify which queue manager it is for.
   - Add to group; use the group that you created
   - Run in a separate Resource Monitor; for better isolation
   - Possible owners; set both nodes
   - Dependencies; add the drive and IP address for this queue manager.
     Warning: Failure to add these dependencies means that WebSphere MQ attempts to write the queue manager status to the wrong cluster disk during failovers. Because many processes might be attempting to write to this disk simultaneously, some WebSphere MQ processes could be blocked from running.
   - Parameters; as follows:
     - QueueManagerName (required); the name of the queue manager that this resource is to control. This queue manager must exist on the local computer.
     - PostOnlineCommand (optional); you can specify a program to run whenever the queue manager resource changes its state from offline to online. For more details see PostOnlineCommand and PreOfflineCommand.
     - PreOfflineCommand (optional); you can specify a program to run whenever the queue manager resource changes its state from online to offline. For more details see PostOnlineCommand and PreOfflineCommand.
       Note: »The *looksAlive* poll interval is set to default value of 5000 ms. The *isAlive* poll interval is set to default value of 30000 ms. These defaults can only be modified after the resource definition has been completed. For further details see Summary of looksAlive and isAlive polling.«
8. Optionally, set a preferred node (but note the comments in Using preferred nodes)
9. The *Failover Policy* (as defined in the properties for the group) is set by default to sensible values, but you can tune the thresholds and periods that control *Resource Failover* and *Group Failover* to match the loads placed on the queue manager.
10. Test the queue manager by bringing it online in the MSCS Cluster Administrator and subjecting it to a test workload. If you are experimenting with a test queue manager, use the WebSphere MQ Explorer. For example:
    a. Right-click the Queues tree node, then select New->Local Queue..., and give the queue a name.
    b. Click Finish. The queue is created, and displayed in the content view.
    c. Right-click the queue, then select Put Test Message.... The Put Test Message panel is displayed.
    d. Type some message text, then click Put Test Message, and close the panel.
    e. Right-click the queue, then select Browse Messages.... The Message Browser panel is displayed.
    f. Ensure that your message is on the queue, then click Close . The Message Browser

panel closes.

g. Right-click the queue, then select Clear Messages.... The messages on the queue are cleared.

h. Right-click the queue, then select Delete.... A confirmation panel is displayed, click OK. The queue is deleted.

11. Test that the queue manager can be taken offline and back online using the MSCS Cluster Administrator.

12. Simulate a failover.
In the MSCS Cluster Administrator, right-click the group containing the queue manager and select Move Group. This can take some minutes to do. (If at other times you want to move a queue manager to another node quickly, follow the procedure in Moving a queue manager to MSCS storage.) You can also right-click and select Initiate Failure; the action (local restart or failover) depends on the current state and the configuration settings.

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14240_

6.6.3.2.6. Summary of looksAlive and isAlive polling

looksAlive and *isAlive* are intervals at which MSCS will call back into the resource types supplied library code and request that the resource performs checks to determine the working status of itself. This ultimately determines if MSCS attempts to fail over the resource.

On every occasion that the *looksAlive* interval elapses (default 5000ms), the queue manager resource is called to perform its own check to determine if its status is satisfactory.

On every occasion that the *isAlive* interval elapses (default 30000ms), another call is made to the queue manager resource for it to perform another check to determine if the resource is functioning correctly. This enables two levels of resource type checking.

1. A *looksAlive* status check to establish if the resource appears to be functioning.
2. A more significant *isAlive* check that determines if the queue manager resource is really active.

If the queue manager resource is determined not to be active, MSCS will, based on other advanced MSCS options, trigger a fail over for the resource and associated dependant resources to another node in the cluster. For further information, refer to the MSCS documentation.

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14241_

6.6.3.2.7. Removing a queue manager from MSCS control

You can remove queue managers from MSCS control, and return them to manual administration. You do not need to do this for maintenance operations. You can do that by taking a queue manager offline temporarily, using the MSCS Cluster Administrator. Removing a queue manager from MSCS control is a more permanent change; only do it if you decide that you no longer want MSCS to have any further control of the queue manager.

If the queue manager being removed uses SSL connections you must modify the queue manager attribute, SSLKEYR, using the WebSphere® MQ Explorer or the MQSC command ALTER QMGR, to point to the SSL key repository file on the local directory.

The procedure is:

1. Take the queue manager resource offline using the MSCS Cluster Administrator. To do this, see Taking a queue manager offline from MSCS.
2. Destroy the resource instance. This does not destroy the queue manager.
3. Optionally, migrate the queue manager files back from shared drives to local drives. To

do this, see Returning a queue manager from MSCS storage.

4. Test the queue manager.

»Taking a queue manager offline from MSCS«

»Returning a queue manager from MSCS storage«

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14250_

6.6.3.2.7.1. Taking a queue manager offline from MSCS

The procedure is:

1. Start the MSCS Cluster Administrator.
2. Open a connection to the cluster.
3. Select Groups, and open the group containing the queue manager to be moved.
4. Select the queue manager resource.
5. Right-click it and select Offline.
6. Wait for completion.

Parent topic: »Removing a queue manager from MSCS control«

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14260_

6.6.3.2.7.2. Returning a queue manager from MSCS storage

This procedure configures the queue manager to be back on its computer's local drive, that is, it becomes a *normal* WebSphere® MQ queue manager. To achieve this, you move the log files and data files from the shared disks. For example, the existing queue manager might have paths such as E:\WebSphere MQ\log\<QMname> and E:\WebSphere MQ\qmgrs\<QMname>. Do not try to move the files by hand; use the hamvmqm utility program supplied as part of WebSphere MQ MSCS Support as described below:

1. Shut down the queue manager, and check that there are no errors.
2. Make a full media backup of the queue files and log files and store the backup in a safe place (see Queue manager log files for why this is important).
3. Decide which local drive to use and ensure that it has sufficient capacity to store the queue manager log files and data (queue) files.
4. Make sure that the shared disk on which the files currently reside is online on the cluster node to which to move the queue manager log and data files.
5. Run the utility program to move the queue manager as follows:

   hamvmqm /m *qmname* /dd "*c*:*WebSphere MQ*" /ld "c:*WebSphere MQ*\log"

   substituting your queue manager name for *qmname*, your local disk drive letter for *c*, and your chosen directory for *WebSphere MQ* (the directories are created if they do not already exist).
6. Test the queue manager to ensure that it works (as described in Moving a queue manager to MSCS storage).

Parent topic: »Removing a queue manager from MSCS control«

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14270_

6.6.3.2.8. Hints and tips on using MSCS

This section contains some general information to help you use WebSphere® MQ support for MSCS effectively.

How long does it take to fail a queue manager over from one machine to the other? This depends heavily on the amount of workload on the queue manager and on the mix of traffic, that is, how much of it is persistent, within sync point, how much committed before the failure, and so on. In our test we have seen failover and failback times of about a minute. This was on a very lightly loaded queue manager and actual times will vary considerably depending on load.

- Verifying that MSCS is working
- Manual startup
- MSCS and queue managers
- Always use MSCS to manage clusters
- Working in Active/Active mode
- PostOnlineCommand and PreOfflineCommand
  Use these commands to integrate WebSphere MQ MSCS support with other systems. You can use them to issue WebSphere MQ commands, wih some restrictions.
- Using preferred nodes
  It can be useful when using Active/Active mode to configure a *preferred node* for each queue manager. However, in general it is better not to set a preferred node but to rely on a manual failback.
- If COM+ errors occur in the Application Event log
  When you install WebSphere MQ on a newly-installed MSCS cluster, you might find an error with Source COM+ and Event ID 4691 reported in the Application Event log.

Parent topic: Supporting the Microsoft Cluster Service (MSCS)

This build: January 26, 2011 11:55:11

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14280_

6.6.3.2.8.1. Verifying that MSCS is working

The task descriptions starting with Creating a queue manager for use with MSCS assume that you have a running MSCS cluster within which you can create, migrate, and destroy resources. If you want to make sure that you have such a cluster:

1. Using the MSCS Cluster Administrator, create a group.
2. Within that group, create an instance of a generic application resource, specifying the system clock (path name C:\winnt\system32\clock.exe and working directory of C:\).
3. Make sure that you can bring the resource online, that you can move the group that contains it to the other node, and that you can take the resource offline.

Parent topic: Hints and tips on using MSCS

This build: January 26, 2011 11:55:11

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14290_

6.6.3.2.8.2. Manual startup

For a queue manager managed by MSCS, you *must* set the startup attribute to manual. This ensures that the WebSphere® MQ MSCS support can restart the IBM® MQSeries® Service without immediately starting the queue manager.

The WebSphere MQ MSCS support needs to be able to restart the service so that it can perform monitoring and control, but must itself remain in control of which queue managers are running, and on which machines. SeeMoving a queue manager to MSCS storage for more information.

Parent topic: Hints and tips on using MSCS

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14310_

6.6.3.2.8.3. MSCS and queue managers

This section describes some things to consider about your queue managers when using MSCS, as follows:

- Creating a matching queue manager on the other node
- Default queue managers
- Deleting a queue manager
- Support for existing queue managers
- Telling MSCS which queue managers to manage
- Queue manager log files
- Multiple queue managers

Creating a matching queue manager on the other node

Default queue managers

Deleting a queue manager

Support for existing queue managers

Telling MSCS which queue managers to manage

Queue manager log files

Multiple queue managers

Parent topic: Hints and tips on using MSCS

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14320_

6.6.3.2.8.3.1. Creating a matching queue manager on the other node

For clustering to work with WebSphere® MQ, you need an identical queue manager on node B for each one on node A. However, you do not need to explicitly create the second one. You can create or prepare a queue manager on one node, move it to the other node as described in Moving a queue manager to MSCS storage, and it is fully duplicated on that node.

Parent topic: MSCS and queue managers

This build: January 26, 2011 11:55:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14330_

6.6.3.2.8.3.2. Default queue managers

Do not use a default queue manager under MSCS control. A queue manager does not have a property that makes it the default; WebSphere® MQ keeps its own separate record. If you move a queue manager set to be the default to the other computer on failover, it does not become the default there. Make all your applications refer to specific queue managers by name.

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:11

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14340_

6.6.3.2.8.3.3. Deleting a queue manager

Once a queue manager has moved node, its details exist in the registry on both computers. When you want to delete it, do so as normal on one computer, and then run the utility described in WebSphere MQ MSCS support utility programs to clean up the registry on the other computer.

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:12

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14350_

6.6.3.2.8.3.4. Support for existing queue managers

You can put an existing queue manager under MSCS control, provided that you can put your queue manager log files and queue files on a disk that is on the shared SCSI bus between the two machines (see Figure 1). You need to take the queue manager offline briefly while the MSCS Resource is created.

If you want to create a new queue manager, create it independently of MSCS, test it, then put it under MSCS control. See:

- Creating a queue manager for use with MSCS
- Moving a queue manager to MSCS storage
- Putting a queue manager under MSCS control

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:12

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14360_

6.6.3.2.8.3.5. Telling MSCS which queue managers to manage

You choose which queue managers are placed under MSCS control by using the MSCS Cluster Administrator to create a resource instance for each such queue manager. This process presents you with a list of resources from which to select the queue manager that you want that instance to manage.

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:12

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14370_

6.6.3.2.8.3.6. Queue manager log files

When you move a queue manager to MSCS storage, you move its log and data files to a shared

disk (for an example see Moving a queue manager to MSCS storage).

It is advisable before you move, to shut the queue manager cleanly and take a full backup of the data files and log files.

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14380_

6.6.3.2.8.3.7. Multiple queue managers

WebSphere MQ MSCS support allows you to run multiple queue managers on each machine and to place individual queue managers under MSCS control.

Parent topic: »MSCS and queue managers«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14390_

6.6.3.2.8.4. Always use MSCS to manage clusters

Do not try to perform start and stop operations directly on any clustered queue manager using either the WebSphere® MQ Explorer. Instead, use the MSCS Cluster Administrator to request that MSCS brings the queue manager online or takes it offline. This is partly to prevent possible confusion caused by MSCS reporting that the queue manager is offline, when in fact you have started it outside the control of MSCS. More seriously, stopping a queue manager without using MSCS is detected by MSCS as a failure, initiating failover to the other node.

Parent topic: »Hints and tips on using MSCS«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14400_

6.6.3.2.8.5. Working in Active/Active mode

Both computers in the MSCS cluster can run queue managers in Active/Active mode. You do not need to have a completely idle machine acting as standby (but you can, if you want, in Active/Passive Mode). If you plan to use both machines to run workload, provide each with sufficient capacity (processor, memory, secondary storage) to run the entire cluster workload at a satisfactory level of performance.

Note: If you are using MSCS together with Microsoft Transaction Server (COM+), you cannot use Active/Active mode. This is because, to use WebSphere® MQ with MSCS and COM+:

- Application components that use WebSphere MQ's COM+ support must run on the same computer as the Distributed Transaction Coordinator (DTC), a part of COM+.
- The queue manager must also run on the same computer.
- The DTC must be configured as an MSCS resource, and can therefore run on only one of the computers in the cluster at any time.

Parent topic: »Hints and tips on using MSCS«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14410_

6.6.3.2.8.6. PostOnlineCommand and PreOfflineCommand

Use these commands to integrate WebSphere® MQ MSCS support with other systems. You can use them to issue WebSphere MQ commands, wih some restrictions.

Specify these commands in the Parameters to a resource of type IBM WebSphere MQ MSCS. You can use them to integrate WebSphere MQ MSCS support with other systems or procedures. For example, you could specify the name of a program that sends a mail message, activates a pager, or generates some other form of alert to be captured by another monitoring system.

»PostOnlineCommand is invoked when the resource changes from offline to online; PreOfflineCommand is invoked for a change from online to offline. When invoked these commands are run, by default, from the Windows system directory. Because WebSphere MQ uses a 32-bit resource monitor process, on Windows 64-bit systems, this is the \Windows\SysWOW64 directory rather than \Windows\system32. For more information, see the Microsoft documentation about file redirection in a Windows x64 environment. Both commands run under the user account used to run the MSCS Cluster Service; and are invoked asynchronously; WebSphere MQ MSCS support does not wait for them to complete before continuing. This eliminates any risk that they might block or delay further cluster operations.«

You can also use these commands to issue WebSphere MQ commands, for example to restart Requester channels. However, the commands are run at the point in time when the queue manager's state changes so they are not intended to perform long-running functions and must not make assumptions about the current state of the queue manager; it is quite possible that, immediately after the queue manager was brought online, an administrator issued an offline command.

If you want to run programs that depend on the state of the queue manager, consider creating instances of the MSCS Generic Application resource type, placing them in the same MSCS group as the queue manager resource, and making them dependent on the queue manager resource.

Parent topic: »Hints and tips on using MSCS«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14420_

6.6.3.2.8.7. Using preferred nodes

It can be useful when using Active/Active mode to configure a *preferred node* for each queue manager. However, in general it is better not to set a preferred node but to rely on a manual failback.

Unlike some other relatively stateless resources, a queue manager can take a while to fail over (or back) from one node to the other. To avoid unnecessary outages, test the recovered node before failing a queue manager back to it. This precludes use of the immediate failback setting. You can configure failback to occur between certain times of day.

Probably the safest route is to move the queue manager back manually to the required node, when you are certain that the node is fully recovered. This precludes use of the preferred node option.

Parent topic: »Hints and tips on using MSCS«

This build: January 26, 2011 11:55:12

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14430_

6.6.3.2.8.8. If COM+ errors occur in the Application Event log

When you install WebSphere® MQ on a newly-installed MSCS cluster, you might find an error with Source COM+ and Event ID 4691 reported in the Application Event log.

This means that you are trying to run WebSphere MQ on a Microsoft Cluster Server (MSCS) environment when the Microsoft Distributed Transaction Coordinator (MSDTC) has not been configured to run in such an environment. For information on configuring MSDTC in a clustered environment, refer to Microsoft documentation.

Parent topic: »Hints and tips on using MSCS«

This build: January 26, 2011 11:56:14

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22790_

6.6.3.2.9. WebSphere MQ MSCS support utility programs

WebSphere® MQ support for MSCS includes the following utility programs that you can run at a command prompt:

Register/unregister the resource type

haregtyp.exe

After you *unregister* the WebSphere MQ MSCS resource type you can no longer create any resources of that type. MSCS does not let you unregister a resource type if you still have instances of that type within the cluster:

1. Using the MSCS Cluster Administrator, stop any queue managers that are running under MSCS control, by taking them offline as described in Taking a queue manager offline from MSCS.
2. Using the MSCS Cluster Administrator, delete the resource instances.
3. At a command prompt, unregister the resource type by entering the following command:

   haregtyp /u

If you want to *register* the type (or re-register it at a later time), enter the following command at a command prompt:

   haregtyp /r

After successfully registering the MSCS libraries, you must reboot the system if you have not done so since installing WebSphere MQ.

Move a queue manager to MSCS storage

hamvmqm.exe

See Moving a queue manager to MSCS storage.

Delete a queue manager from a node

hadltmqm.exe

Consider the case where you have had a queue manager in your cluster, it has been moved from one node to another, and now you want to destroy it. Use the WebSphere MQ Explorer to delete it on the node where it currently is. The registry entries for it still exist on the other computer. To delete these, enter the following command at a prompt on that computer:

   hadltmqm /m qmname

where qmname is the name of the queue manager to remove.

Check and save setup details

amqmsysn.exe

This utility presents a dialog showing full details of your WebSphere MQ MSCS Support setup, such as might be requested if you call IBM® support. There is an option to save the details to a file.

Parent topic: »Supporting the Microsoft Cluster Service (MSCS)«

This build: January 26, 2011 11:55:12

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14450_

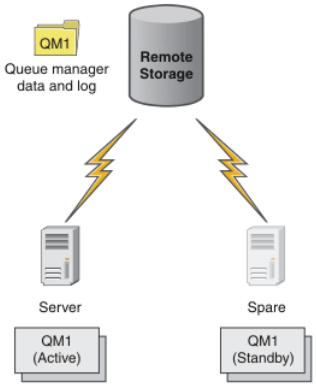
»6.6.3.3. Multi-instance« queue managers

A »multi-instance« queue manager restarts automatically on a standby server.

Figure 1 shows a »multi-instance« configuration for QM1. WebSphere® MQ is installed on two servers, one of which is a spare. One queue manager, QM1, has been created. One instance of QM1 is active, and is running on one server. The other instance of QM1 is running in standby on the other server, doing no active processing, but ready to take over from the active

instance of QM1, if the active instance fails.

Figure 1. Multi-instance queue manager



When you intend to use a queue manager as a multi-instance queue manager, create a single queue manager on one of the servers using the crtmqm command, placing its queue manager data and logs in shared network storage. On the other server, rather than create the queue manager again, use the addmqinf command to create a reference to the queue manager data and logs on the network storage.

You can now run the queue manager from either of the servers. Each of the servers references the same queue manager data and logs; there is only one queue manager, and it is active on only one server at a time.

The queue manager can run either as a single instance queue manager, or as a multi-instance queue manager. In both cases only one instance of the queue manager is running, processing requests. The difference is that when running as a multi-instance queue manager, the server that is not running the active instance of the queue manager runs as a standby instance, ready to take over from the active instance automatically if the active server fails.

The only control you have over which instance becomes active first is the order in which you start the queue manager on the two servers. The first instance to acquire read/write locks to the queue manager data becomes the active instance.

You can swap the active instance to the other server, once it has started, by stopping the active instance using the switchover option to transfer control to the standby.

The active instance of QM1 has exclusive access to the shared queue manager data and logs folders when it is running. The standby instance of QM1 detects when the active instance has failed, and becomes the active instance. It takes over the QM1 data and logs in the state they were left by the active instance, and accepts reconnections from clients and channels.

The active instance might fail for various reasons that result in the standby taking over:

- Failure of the server hosting the active queue manager instance.
- Failure of connectivity between the server hosting the active queue manager instance and the file system.
- Unresponsiveness of queue manager processes, detected by WebSphere MQ, which then shuts down the queue manager.

You can add the queue manager configuration information to multiple servers, and choose any two servers to run as the active/standby pair.

A »multi-instance« queue manager is one part of a high availability solution. You need some additional components to build a useful high availability solution.

- Client and channel reconnection to transfer WebSphere MQ connections to the computer that takes over running the active queue manager instance.
- A high performance shared network file system that manages locks correctly and provides protection against media and file server failure.
- Resilient networks and power supplies to eliminate single points of failure in the basic infrastructure.
- Applications that tolerate failover. In particular you need to pay close attention to the behavior of transactional applications, and to applications that browse WebSphere MQ queues.
- Monitoring and management of the active and standby instances to ensure that they are running, and to restart active instances that have failed. Although »multi-instance« queue managers restart automatically, you need to be sure that your standby instances are running, ready to take over, and that failed instances are brought back online as new

standby instances.

WebSphere MQ Clients and channels reconnect automatically to the standby queue manager when it becomes active. Reconnection, and the other components in a high availability solution are discussed in related topics. Automatic client reconnect is not supported by WebSphere MQ classes for Java.

Supported platforms

You can create a »multi-instance« queue manager on any of the non-z/OS platforms from release v7.0.1.

Automatic client reconnection is supported for MQI clients from release v7.0.1 onwards.

»Creating a multi-instance queue manager«
Create a »multi-instance« queue manager by creating the queue manager on one server, and configuring WebSphere MQ on another server to use the shared queue manager data and logs.

»Deleting a multi-instance queue manager«
To delete a multi-instance queue manager completely, you need to use the dltmqm command to delete the queue manager, and then remove instances from other servers using either the rmvmqinf or dltmqm commands.

»Starting and stopping a multi-instance queue manager«
Starting and stopping a queue manager configured either as a »single instance« or a »multi-instance« queue manager.

»Shared file system«
A »multi-instance« queue manager uses a networked file system to manage queue manager instances.

»Multiple queue manager instances«
A »multi-instance« queue manager is resilient because it uses a standby queue manager instance to restore queue manager availability after failure.

»Failover or switchover«
A standby queue manager instance takes over from the active instance either on request (switchover), or when the active instance fails, (failover).

»Channel and client reconnection«
Channel and client reconnection is an essential part of restoring message processing after a standby queue manager instance has become active.

»Application recovery«
Application recovery is the automated continuation of application processing after failover. Application recovery following failover requires careful design. Some applications need to be aware failover has taken place.

»Data recovery and high availability«
High availability solutions using »multi-instance« queue managers must include a mechanism to recover data after a storage failure.

»Combining WebSphere MQ Availability solutions«
Applications are using other WebSphere MQ capabilities to improve availability. »Multi-instance« queue managers complement other high availability capabilities.

Parent topic: »Using WebSphere MQ with high availability configurations«

Related concepts
»File system support«
»Automatic client reconnection«

This build: January 26, 2011 11:56:17

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
highlyavailablequeuemanagers
»

6.6.3.3.1. Creating a »multi-instance« queue manager

Create a »multi-instance« queue manager by creating the queue manager on one server, and configuring WebSphere® MQ on another server to use the shared queue manager data and logs.

Most of the work of setting up a »multi-instance« queue manager involves setting up the network storage that holds the queue manager data and log files, and making the files available to other servers using network shares. These tasks need to be performed by someone with administrative authority, such as *root* on UNIX systems. Once the shares are set up, and a normal queue manager has been created using the shares for its queue manager data and logs, you need only configure WebSphere MQ on the other servers. You do not create the queue manager again on the other servers.

File access control

You need to take care that the user and group mqm on all other servers have permission to access the shares.

On UNIX, you need to make the uid and gid of mqm the same on all the systems. You might need to edit /etc/passwd on each system to set a common uid and gid for mqm, and then reboot your UNIX systems.

On Microsoft Windows, you must install WebSphere MQ on a domain server, and create a user who is a member of the global group domain mqm to own WebSphere MQ resources. Make this user the owner of the shared queue manager and log files. The sid of the user who owns the queue manager and log file is then the same as the sid of the user that runs instances of the queue manager.

Configuration information

Configure as many queue manager instances as you need by modifying the WebSphere MQ configuration information about each server that has WebSphere MQ installed and is to run a queue manager instance. The commands, dspmqinf and addmqinf help you to configure the additional queue manager instances, or you can edit the mqs.ini file on Unix servers directly. The topics, Create a multi-instance queue manager on Linux and Create a multi-instance queue manager on Windows Server are examples showing how to configure a multi-instance queue manager.

On UNIX systems, you can share a single mqs.ini file by placing it on the network share and setting the AMQ_MQS_INI_LOCATION environment variable to point to it.

On Microsoft Windows, WebSphere MQ has for a number of releases kept its configuration information in the Windows registry. In the 7.0.1 release some configuration information is moved from the registry into configuration files.

Multi-instance queue manager configuration is moved out of the registry into qm.ini and qmstatus.ini files, which are located in the queue manager data directory. The WebSphere MQ configuration information (mqs.ini on UNIX platforms) remains in the registry on Windows as does all the configuration information for existing queue managers and new queue managers that are configured to use the default data directory.

Restrictions

1. Configure multiple instances of the same queue manager only on servers having the same operating system, architecture (both machines having 32 bit or 64 bit word size, for example), and endianness.
2. All WebSphere MQ installations must be at release level 7.0.1 or higher.
3. »Typically active and standby installations are maintained at the same maintenance level. Consult the maintenance instructions for each upgrade to check if you must upgrade all installations together.«
4. The network share that contains queue manager data and logs can only be shared between queue managers that are configured with the same MQ user, group, and access control mechanism. For example, the network share set up on a Linux server could contain separate queue manager data and logs for AIX®, Solaris, HP-UX and Linux queue managers, but could not contain the queue manager data used by IBM® i.
   You can create multiple shares on the same networked storage for IBM i and other UNIX systems as long as the shares are different. You can give different shares different owners. The restriction is a consequence of the different names used for the WebSphere MQ user and group between UNIX and IBM i, although the user and group can have the same uid and gid.
5. On UNIX systems, configure the shared file system on networked storage with a hard, interruptible, mount rather than a soft mount. A hard interruptible mount forces the queue manager to hang until it is interrupted by a system call. Soft mounts do not

guarantee data consistency after a server crash.

6. Microsoft Windows shared log and data directories must be owned by the domain mqm group set up on a Windows Domain Server.
7. The shared log and data directories cannot be stored on a FAT, or an NFSv3 file system.
8. z/OS® does not support multi-instance queue managers. Use queue sharing groups. Reconnectable clients do work with z/OS queue managers.

»Create a multi-instance queue manager on Windows Server«
An example shows how to set up an example »multi-instance« queue manager on Windows Server. The setup is small and simple, to demonstrate the concepts involved, rather than being production scale. The example is based on Windows Server 2003. The steps might differ on other versions of Windows Server.

»Create a multi-instance queue manager on Linux«
An example shows how to set up a »multi-instance« queue manager on Linux. The setup is small to illustrate the concepts involved. The example is based on Linux Red Hat Enterprise 5. The steps differ on other UNIX platforms.

Parent topic: »Multi-instance queue managers«

Related reference
addmqinf
dspmqinf
rmvmqinf

This build: January 26, 2011 11:56:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70159_
«

6.6.3.3.1.1. Create a multi-instance queue manager on Windows Server

An example shows how to set up an example »multi-instance« queue manager on Windows Server. The setup is small and simple, to demonstrate the concepts involved, rather than being production scale. The example is based on Windows Server 2003. The steps might differ on other versions of Windows Server.

Before you begin

On Windows, multi-instance queue managers must run on a domain controller. For this reason, and for this example, install two Windows servers, and configure them as domain controllers in the same domain. The topic, Windows domains and multi-instance queue managers explains why multi-instance queue managers must run on a domain controller, and how to configure users and groups to run a multi-instance queue manager on a domain controller.

The example was set up on a 2 GHz notebook computer with 3 GB RAM running Windows XP Service Pack 2. Two VMware virtual machines ran Windows Server 2003 Service Pack 2 in 500 MB images. A WebSphere MQ client was installed on an additional 400 MB VMware image that ran Windows XP Service Pack 2 to run the sample high availability applications. All the virtual machines were configured as part of a VMware host-only network for security reasons. Although it is recommended not to use automatically generated IP address, for a short demonstration it is satisfactory as the IP addresses ought not to change. As part of the configuration you need to update the TCP/IP properties of the virtual machines with the address of the domain server you have chosen to be the DNS server.

If you configure a queue manager to run as a service, you need to check that the user the service logs on as is a member of the local group mqm.

Example

Table 1. Illustrative »multi-instance« queue manager configuration on Windows Server 2003

Server 1	Server 2
Log in with user who is a member of the local group mqm.	

Create log and data directories in a folder on an NTFS drive, c:\MQHA, making sure that the owner is a member of mqm, and mqm has full-control authority to the folders. C:\MQHA\data C:\MQHA\log	
Create a share MQHA for C:\MQHA.	Connect to *hostname*\MQHA
Using WebSphere MQ Explorer, create a queue manager, QM1, overriding the default paths to the data and log folders and use UNC names to refer to the data and log folders. *hostname*\MQHA\data and *hostname*\MQHA\log.	
On a command-line run, **dspmqinf -o** command QM1 and copy the result to the clip board, addmqinf -s QueueManager **-v** Name=QM1 **-v** Directory=QM1 **-v** Prefix="C:\IBM\MQ\Data" **-v** DataPath="*hostname*\MQHA\data\QM1"	Paste and run the command, **addmqinf -s** QueueManager **-v** Name=QM1 **-v** Directory=QM1 **-v** Prefix="C:\IBM\MQ\Data" **-v** DataPath="*hostname*\MQHA\data\QM1"
Using WebSphere® MQ Explorer, start the manager instances with the **Permit a standby instance** check box selected.	

»Windows domains and multi-instance queue managers«
Multi-instance queue managers on Windows require the local mqm group to be a domain local group.

»Verify the multi-instance queue manager on Windows«
Use the sample programs amqsghac, amqsphac and amqsmhac to verify a multi-instance queue manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Windows Server 2003.

Parent topic: »Creating a multi-instance queue manager«

Related information
Creating and setting up domain accounts on WebSphere MQ

This build: January 26, 2011 11:56:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70161_

6.6.3.3.1.1.1. Windows domains and multi-instance queue managers

Multi-instance queue managers on Windows require the local mqm group to be a domain local group.

When WebSphere MQ is installed on Windows, it creates a local group, mqm. The local mqm group is used for access control. When the queue manager accesses its queue manager data and log directories it checks that the user it is running under is a member of the local mqm group, that the owner of the directories is also a member of the same local mqm group, and the local mqm group has read/write permissions to the queue manager directories. If the check fails, that is it. The queue manager does not go on to check any domain mqm group permissions that might have been given as well.

Multi-instance queue managers require their data and log directories to be on shared network storage, on a different server for reliability reasons. The result is, when you configure the file server, and the two queue manager servers using their local mqm groups, that you have three different local mqm groups, which have different SIDs. When the queue manager makes its access control checks, the checks fail.

The only way to ensure each of the servers running queue manager instances use the same local mqm group with the same SID as the owner of the queue manager data and log directories on the file server is to make the local mqm group a *domain local group*.

The user who creates the queue manager data and log directories on the file server can now be a member of the same domain local group mqm, as the user who runs one of the queue manager instances, and the user who runs the other queue manager instance.

By using the same domain local group mqm, when the queue manager checks its permission to access the data and log directories, the SID of the local mqm group of the queue manager matches the SID of the local mqm group of the user that created the directories, and permission is granted.

In order to use *domain local groups*, you must run multi-instance queue managers on a *domain controller*. On a domain controller all local groups are implicitly domain local groups.

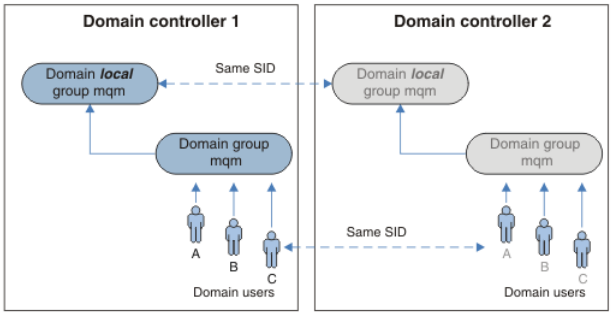
On each domain controller the local mqm group has the same SID as the local mqm group on another domain controller in the same domain. Each local mqm group really is the same domain local group mqm.

You can create and connect as many domain controllers as you need.

Suggested multi-instance queue manager configuration

Figure 1 shows the suggested configuration of the users and groups for a multi-instance queue manager on Windows. The grayed out text in the figure indicates the domains and users are references and not different objects. The references therefore have the same SIDs as the actual definitions.

Figure 1. Using a domain local group to configure a multi-instance queue manager

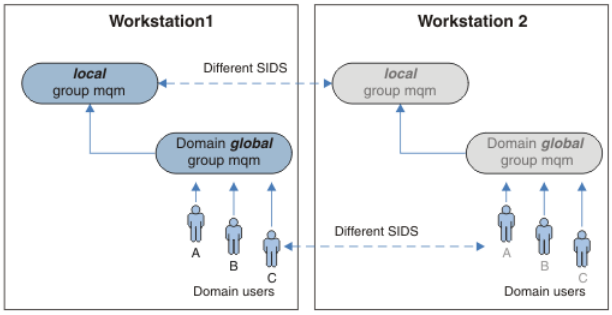


By creating the users members of the domain group mqm, rather than of the domain local group mqm, you only need to define each user once for all the servers in the domain. The name of the domain group mqm has no significance to the queue manager.

Invalid configuration of multi-instance queue manager

Figure 2 shows a typical configuration of WebSphere MQ on Windows. The configuration does not work with multi-instance queue managers because the two mqm local groups have different SIDs. They are different groups.

Figure 2. Incorrect configuration of a multi-instance queue manager



Parent topic: »Create a multi-instance queue manager on Windows Server«

This build: January 26, 2011 11:56:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70169_

6.6.3.3.1.1.2. Verify the multi-instance queue manager on Windows

Use the sample programs amqsghac, amqsphac and amqsmhac to verify a multi-instance queue

manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Windows Server 2003.

The high availability sample programs are designed to use automatic client reconnection. When the connected queue manager fails, the client attempts to reconnect to a queue manager in the same queue manager group. The description of the samples, High availability sample programs, demonstrates client reconnection using a single instance queue manager for simplicity. You can use the same samples with multi-instance queue managers to verify a multi-instance queue manager configuration.

This example uses the multi-instance configuration described in Create a multi-instance queue manager on Windows Server. Use the configuration to verify that the multi-instance queue manager switches over to the standby instance. Either stop the queue manager in some catastrophic way, or end it with the endmqm command and use the »-s«, switchover, option. The client programs reconnect to the new queue manager instance and continue to work with the new instance after a slight delay.

The client is installed in a 400 MB VMware image that is running Windows XP Service Pack 2. For security reasons, it is connected on the same VMware host-only network as the domain servers that are running the multi-instance queue manager. It is sharing the /MQHA folder, which contains the client connection table, to simplify configuration.

Verifying failover using WebSphere MQ Explorer

Before using the sample applications to verify failover, run the WebSphere MQ Explorer on each server. Add both queue manager instances to each explorer using the Add Remote Queue Manager > Connect directly to a multi-instance queue manager wizard. Ensure that both instances are running, permitting standby. Close the window running the VMware image with the active instance, virtually powering off the server, or stop the active instance, allowing switchover to standby instance and reconnectable clients to reconnect.

Note: If you power off the server, make sure that it is not the one hosting the MQHA folder!

Note: The Allow switchover to a standby instance option might not be available on the Stop Queue Manager dialog. The option is missing because the queue manager is running as a single instance queue manager. You must have started it without the Permit a standby instance option. If your request to stop the queue manager is rejected, look at the Details window, possibly there is no standby instance running.

Verifying failover using the sample programs

Choose a server to run the active instance

You might have chosen one of the servers to host the MQHA directory or file system. If you plan to test failover by closing the VMware window running the active server, make sure that it is not the one hosting MQHA!

On the server running the active queue manager instance

1. Modify *ipaddr1* and *ipaddr2* and save the following commands in N:\hasample.tst.

```
DEFINE QLOCAL(SOURCE) REPLACE
DEFINE QLOCAL(TARGET) REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)    TRPTYPE(TCP) +
        MCAUSER(' ') REPLACE
DEFINE CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)   TRPTYPE(TCP) +
        CONNAME('ipaddr1(1414),ipaddr2(1414)') QMNAME(QM1) REPLACE
START  CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

   Note: By leaving the MCAUSER parameter blank, the client user ID is sent to the server. The client user ID must have the correct permissions on the servers. An alternative is to set the MCAUSER parameter in the SVRCONN channel to the user ID you have configured on the server.

2. Open a command prompt with the path N:\ and run the command:

```
runmqsc -m QM1 < hasample.tst
```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the runmqsc command.

```
LISTENER(LISTENER.TCP)CONTROL(QMGR)
LISTENER(LISTENER.TCP)STATUS(RUNNING)
```

   Or, using the WebSphere MQ Explorer that the TCPIP listener is running and has Control = Queue Manager.

On the client

1. Map the shared directory C:\MQHA on the server to N:\ on the client.
2. Open a command prompt with the path N:\ . Set the environment variable MQCHLLIB to point to the client channel definition table (CCDT) on the server:

   ```
   SET MQCHLLIB=N:\data\QM1\@ipcc
   ```

3. At the command prompt type the commands:

   ```
   start amqsghac TARGET QM1
   start amqsmhac -s SOURCE -t TARGET -m QM1
   start amqsphac SOURCE QM1
   ```

   Note: If you have problems, start the applications at a command prompt so that the reason code is printed out on the console, or look at the AMQERR01.LOG file in the N:\data\QM1\errors folder.

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the WebSphere MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing reconnectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be re-established. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

Parent topic: »Create a multi-instance queue manager on Windows Server«

This build: January 26, 2011 11:56:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70163_

6.6.3.3.1.2. Create a multi-instance queue manager on Linux

An example shows how to set up a »multi-instance« queue manager on Linux. The setup is small to illustrate the concepts involved. The example is based on Linux Red Hat Enterprise 5. The steps differ on other UNIX platforms.

The example is set up on a 2 GHz notebook computer with 3 GB RAM running Windows XP Service Pack 2. Two VMware virtual machines run Linux Red Hat Enterprise 5 in 640 MB images. A WebSphere MQ client is installed on an additional 400 MB VMware image that runs Windows XP Service Pack 2 and runs the sample high availability applications. All the virtual machines are configured as part of a VMware host-only network for security reasons. Although it is generally recommended not to use an automatically generated IP address, it is acceptable to use one for a short demonstration, because the IP address is unlikely to change.

Example

Table 1. Illustrative »multi-instance« queue manager configuration on Linux

Server 1	Server 2
Log in as *root*	
Follow the instructions in Quick Beginnings to install WebSphere® MQ, create the mqm user and group, and define /var/mqm.	
Carry out the task, Verifying shared file system locking to check the file system supports multi-instance queue managers.	
Check what uid and gid /etc/passwd displays for mqm, for example, mqm:x:501:100:MQ User:/var/mqm:/bin/bash	Match the uid and gid for mqm in /etc/passwd and reboot if you have had to change the values.
Create log and data directories in a common folder, /MQHA, that is to be shared. For example, 1. **mkdir** */MQHA* 2. **mkdir** */MQHA/logs* 3. **mkdir** */MQHA/qmgrs*	Create the folder, /MQHA, to mount the shared file system. Keep the path the same as on server 1; for example: 1. **mkdir** */MQHA*
Ensure that the MQHA directories are owned by user and group mqm, and the access permissions are set to rwx for user and group; for example **ls -al** displays, drwxrwxr-x mqm mqm 4096 Nov 27 14:38 MQDATA 1. **chown -R** mqm:mqm */MQHA* 2. **chmod -R** ug+rwx */MQHA*	
Create the queue manager: **crtmqm -ld** */MQHA/logs* **-md** */MQHA/qmgrs* **-q** *QM1*	
Add[1] /MQHA *(rw,sync,no_wdelay,fsid=0) to /etc/exports	
Start the NFS daemon: */etc/init.d/***nfs** start	
Discover the host name or IP address of server 1: **ifconfig**[2]	Mount the exported file system /MQHA: »**mount -t** nfs4 **-o** hard,intr *192.168.217.130:/ /MQHA*«
Copy the queue manager configuration details from Server 1, **dspmqinf -o** command QM1 and copy the result to the clip board, addmqinf -s QueueManager -v Name=QM1 -v Directory=QM1 -v Prefix=/var/mqm -v DataPath=/MQHA/qmgrs/QM1	Paste the queue manager configuration command into Server 2, **addmqinf -s** QueueManager **-v** Name=QM1 **-v** Directory=QM1 **-v** Prefix=/var/mqm **-v** DataPath=/MQHA/qmgrs/QM1
Start the queue manager instances, in either order, with the **-x** parameter: **strmqm -x** QM1	

»Verifying the multi-instance queue manager on Linux«
Use the sample programs amqsghac, amqsphac and amqsmhac to verify a multi-instance queue manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Linux Red Hat Enterprise 5.

Parent topic: »Creating a multi-instance queue manager«

[1] The '*' allows all machines that can reach this one mount /MQHA for read/write. Restrict access on a production machine.

[2] Try the command /sbin/ifconfig if the simple ifconfig command does not work.

This build: January 26, 2011 11:56:20

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70160_

6.6.3.3.1.2.1. Verifying the multi-instance queue manager on Linux

Use the sample programs amqsghac, amqsphac and amqsmhac to verify a multi-instance queue manager configuration. This topic provides an example configuration to verify a multi-instance queue manager configuration on Linux Red Hat Enterprise 5.

The high availability sample programs use automatic client reconnection. When the connected queue manager fails, the client attempts to reconnect to a queue manager in the same queue manager group. The description of the samples, High availability sample programs, demonstrates client reconnection using a single instance queue manager for simplicity. You can use the same samples with multi-instance queue managers to verify a multi-instance queue manager configuration.

The example uses the multi-instance configuration described in Create a multi-instance queue manager on Linux. Use the configuration to verify that the multi-instance queue manager switches over to the standby instance. Either stop the queue manager in some catastrophic way, or end it with the endmqm command and use the »-s«, switchover, option. The client programs reconnect to the new queue manager instance and continue to work with the new instance after a slight delay.

In the example, the client is running on a Windows XP Service Pack 2 system. The system is hosting two VMware Linux servers that are running the multi-instance queue manager.

Verifying failover using WebSphere MQ Explorer

Before using the sample applications to verify failover, run the WebSphere MQ Explorer on each server. Add both queue manager instances to each explorer using the Add Remote Queue Manager > Connect directly to a multi-instance queue manager wizard. Ensure that both instances are running, permitting standby. Close the window running the VMware image with the active instance, virtually powering off the server, or stop the active instance, allowing switchover to standby instance.

Note: If you power off the server, make sure that it is not the one hosting /MQHA!

Note: The Allow switchover to a standby instance option might not be available on the Stop Queue Manager dialog. The option is missing because the queue manager is running as a single instance queue manager. You must have started it without the Permit a standby instance option. If your request to stop the queue manager is rejected, look at the Details window, it is possibly because there is no standby instance running.

Verifying failover using the sample programs

Choose a server to be to run the active instance

You might have chosen one of the servers to host the MQHA directory or file system. If you plan to test failover by closing the VMware window running the active server, make sure that it is not the one hosting MQHA!

On the server running the active queue manager instance

Note: Running the SVRCONN channel with the MCAUSER set to mqm, is a convenience to reduce the number of configuration steps in the example. If another user ID is chosen, and your system is set up differently to the one used in the example, you might experience access permission problems. Do not use mqm as a MCAUSER on an exposed system; it is likely to compromise security greatly.

1. Modify *ipaddr1* and *ipaddr2* and save the following commands in /MQHA/hasamples.tst.

```
DEFINE  QLOCAL(SOURCE) REPLACE
DEFINE  QLOCAL(TARGET) REPLACE
DEFINE  CHANNEL(CHANNEL1) CHLTYPE(SVRCONN)    TRPTYPE(TCP) +
         MCAUSER('mqm') REPLACE
DEFINE  CHANNEL(CHANNEL1) CHLTYPE(CLNTCONN)   TRPTYPE(TCP) +
         CONNAME('ipaddr1(1414),ipaddr2(1414)') QMNAME(QM1) REPLACE
START   CHANNEL(CHANNEL1)
DEFINE LISTENER(LISTENER.TCP) TRPTYPE(TCP) CONTROL(QMGR)
DISPLAY LISTENER(LISTENER.TCP) CONTROL
DISPLAY LSSTATUS(LISTENER.TCP) STATUS
```

2. Open a terminal window with the path /MQHA and run the command:

   ```
   runmqsc -m QM1 < hasamples.tst
   ```

3. Verify that the listener is running and has queue manager control, either by inspecting the output of the runmqsc command.

   ```
   LISTENER(LISTENER.TCP)CONTROL(QMGR)
   LISTENER(LISTENER.TCP)STATUS(RUNNING)
   ```

   Or, using the WebSphere MQ Explorer that the TCPIP listener is running and has Control = Queue Manager.

On the client

1. Copy the client connection table AMQCLCHL.TAB from /MQHA/qmgrs/QM1.000/@ipcc on the server to C:\ on the client.
2. Open a command prompt with the path C:\ and set the environment variable MQCHLLIB to point to the client channel definition table

   ```
   SET MQCHLLIB=C:\
   ```

3. At the command prompt type the commands:

   ```
   start amqsghac TARGET QM1
   start amqsmhac -s SOURCE -t TARGET -m QM1
   start amqsphac SOURCE QM1
   ```

On the server running the active queue manager instance

1. Either:
   - Close the window running the VMware image with the active server instance.
   - Using the WebSphere MQ Explorer, stop the active queue manager instance, allowing switchover to the standby instance and instructing reconnectable clients to reconnect.
2. The three clients eventually detect the connection is broken, and then reconnect. In this configuration, if you close the server window, it is taking about seven minutes for all three connections to be re-established. Some connections are reestablished well before others.

Results

```
N:\>amqsphac SOURCE QM1
Sample AMQSPHAC start
target queue is SOURCE
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

```
N:\>amqsmhac -s SOURCE -t TARGET -m QM1
Sample AMQSMHA0 start

17:05:25 : EVENT : Connection Reconnecting (Delay: 97ms)
17:05:48 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:53 : EVENT : Connection Reconnected
```

```
N:\>amqsghac TARGET QM1
Sample AMQSGHAC start
message <Message 1>
message <Message 2>
message <Message 3>
message <Message 4>
message <Message 5>
17:05:25 : EVENT : Connection Reconnecting (Delay: 156ms)
17:05:47 : EVENT : Connection Reconnecting (Delay: 0ms)
17:05:52 : EVENT : Connection Reconnected
message <Message 6>
message <Message 7>
message <Message 8>
message <Message 9>
```

Parent topic: »Create a multi-instance queue manager on Linux«

This build: January 26, 2011 11:56:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70164_

6.6.3.3.2. Deleting a multi-instance queue manager

To delete a multi-instance queue manager completely, you need to use the dltmqm command to delete the queue manager, and then remove instances from other servers using either the rmvmqinf or dltmqm commands.

Run the dltmqm command to delete a queue manager that has instances defined on other servers, on any server where that queue manager is defined. You do not need to run the dltmqm command on the same server that you created it on. Then run the rmvmqinf or dltmqm command on all the other servers which have a definition of the queue manager.

You can only delete a queue manager when it is stopped. At the time you delete it no instances are running, and the queue manager, strictly speaking, is neither a single or a multi-instance queue manager; it is simply a queue manager that has its queue manager data and logs on a remote share. When you delete a queue manager, its queue manager data and logs are deleted, and the queue manager stanza is removed from the mqs.ini file on the server on which you issued the dltmqm command. You need to have access to the network share containing the queue manager data and logs when you delete the queue manager. On Windows there is no mqs.ini file; instead, the queue manager stanza is removed from the registry.

On other Unix servers where you have previously created instances of the queue manager there are also entries in the mqs.ini files on those servers. You need to visit each server in turn, and remove the queue manager stanza by running the command rmvmqinf *Queue manager stanza name*. On Windows, run the same command, and the queue manager stanza is removed from the registry.

On Unix, if you have placed a common mqs.ini file in network storage and referenced it from all the servers by setting the AMQ_MQS_INI_LOCATION environment variable on each server, then you need to delete the queue manager from only one of its servers as there is only one mqs.ini file to update.

Example

First server

```
dltmqm QM1
```

Other servers where instances are defined

```
rmvmqinf QM1, or
dltmqm QM1
```

Parent topic: »Multi-instance queue managers«

This build: January 26, 2011 11:56:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70162_

6.6.3.3.3. Starting and stopping a »multi-instance« queue manager

Starting and stopping a queue manager configured either as a »single instance« or a »multi-instance« queue manager.

When you have defined a »multi-instance« queue manager on a pair of servers, you can run the queue manager on either server, either as a »single instance« queue manager, or as a »multi-instance« queue manager.

To run a »multi-instance« queue manager, start the queue manager on one of the servers using the strmqm -x *QM1* command; the -x option permits the instance to failover. It becomes the *active instance*. Start the standby instance on the other server using the same strmqm -x *QM1* command; the -x option permits the instance to start as a standby.

The queue manager is now running with one active instance that is processing all requests, and a standby instance that is ready to take over if the active instance fails. The active instance is granted exclusive access to the queue manager data and logs. The standby waits to be granted exclusive access to the queue manager data and logs. When the standby is granted exclusive access, it becomes the active instance.

You can also manually switch control to the standby instance by issuing the endmqm -s command on the active instance. The endmqm -s command shuts down the active instance without shutting down the standby. The exclusive access lock on the queue manager data and logs is released, and the standby takes over.

»You can also start and stop a queue manager configured with multiple instances on different servers as a single instance queue manager. If you start the queue manager without using the -x option on the strmqm command, the instances of the queue manager configured on other machines are prevented from starting as standby instances. If you attempt to start another instance you receive the response that the queue manager instance is not permitted to run as a standby.«

»If you stop the active instance of a »multi-instance« queue manager using the endmqm command without the -s option, then the active and standby instances both stop. If you stop the standby instance using the endmqm command with the -x option, then it stops being a standby, and the active instance continues running. You cannot issue endmqm without the -x option on the standby. «

Note: Only two queue manager instances can run at the same time; one is the active instance, and the other is a standby instance. If you start two instances at the same time, WebSphere® MQ has no control over which instance becomes the active instance; it is determined by the network file system. The first instance to acquire exclusive access to the queue manager data becomes the active instance.

Parent topic: »Multi-instance queue managers«

This build: January 26, 2011 11:56:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70151_

6.6.3.3.4. Shared file system

A »multi-instance« queue manager uses a networked file system to manage queue manager instances.

A »multi-instance« queue manager automates failover using a combination of file system locks and shared queue manager data and logs. Only one instance of a queue manager can have exclusive access to the shared queue manager data and logs. When it gets access it becomes the active instance. The other instance that does not succeed in getting exclusive access waits as a standby instance until the queue manager data and logs become available.

The networked file system is responsible for releasing the locks it holds for the active queue manager instance. If the active instance fails in some way, the networked file system releases the locks it is holding for the active instance. As soon as the exclusive lock is released, a standby queue manager waiting for the lock attempts to acquire it. If it succeeds, it becomes the active instance and has exclusive access to the queue manager data and logs on the shared file system. It then continues starting up.

The related topic, File system support describes how to set up and check that your file system supports »multi-instance« queue managers.

A »multi-instance« queue manager does not protect you against a failure in the file system. There are a number of ways to protect your data.

- Invest in reliable storage, such as redundant disk arrays (RAID), and include them in a networked file system that has network resilience.
- Back up WebSphere® MQ linear logs to alternative media, and if your primary log media fails, recover using the logs on the alternative media. You can use a backup queue manager to administer this process.

Parent topic: »Multi-instance queue managers«

Related concepts
Example directory configurations on UNIX systems
»Data recovery and high availability«
»File system support«

This build: January 26, 2011 11:56:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70152_

6.6.3.3.5. Multiple queue manager instances

A »multi-instance« queue manager is resilient because it uses a standby queue manager instance to restore queue manager availability after failure.

Replicating queue manager instances is a very effective way to improve the availability of queue manager processes. Using a simple availability model, purely for illustration: if the reliability of one instance of a queue manager is 99% (over one year, cumulative downtime is 3.65 days) then adding another instance of the queue manager increases the availability to 99.99% (over one year, cumulative downtime of about an hour).

This is too simple a model to give you practical numerical estimates of availability. To model availability realistically, you need to collect statistics for the mean time between failures (MTBF) and the mean time to repair (MTTR), and the probability distribution of time between failures and of repair times.

The term, »multi-instance« queue manager, refers to the combination of active and standby instances of the queue manager that share the queue manager data and logs. »Multi-instance« queue managers protect you against the failure of queue manager processes by having one instance of the queue manager active on one server, and another instance of the queue manager on standby on another server, ready to take over automatically should the active instance fail.

Parent topic: »Multi-instance queue managers«

This build: January 26, 2011 11:56:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70153_

6.6.3.3.6. Failover or switchover

A standby queue manager instance takes over from the active instance either on request (switchover), or when the active instance fails, (failover).

- *Switchover* takes place when a standby instance starts in response to the endmqm -s command being issued to the active queue manager instance. You can specify the endmqm parameters -c, -i or -p to control how abruptly the queue manager is stopped.
  Note: Switchover only takes place if a standby queue manager instance is already started. The endmqm -s command releases the active queue manager lock and permits switchover: it does not start a standby queue manager instance.
- *Failover* occurs when the lock on queue manager data held by the active instance is released due the instance appearing to stop unexpectedly (in other words without an endmqm command being issued).

When the standby instance takes over as the active instance, it writes a message to the queue manager error log.

Reconnectable clients are automatically reconnected when a queue manager fails or switches over. You do not need to include the -r flag on the endmqm command to request client reconnection. Automatic client reconnect is not supported by WebSphere® MQ classes for Java.

Parent topic: »Multi-instance queue managers«

This build: January 26, 2011 11:56:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70154_

6.6.3.3.7. Channel and client reconnection

Channel and client reconnection is an essential part of restoring message processing after a standby queue manager instance has become active.

»Multi-instance« queue manager instances are installed on servers with different network addresses. You need to configure WebSphere® MQ channels and clients with connection information for all queue manager instances. When a standby takes over, clients and channels are automatically reconnected to the newly active queue manager instance at the new network address. Automatic client reconnect is not supported by WebSphere MQ classes for Java.

The design is different to the way high availability environments such as HA-CMP provide a virtual IP address for the cluster and transfer the address to the active server. WebSphere MQ reconnection does not change or reroute IP addresses. It works by reconnecting using the network addresses you have defined in channel definitions and client connections. As an administrator, you need to define the network addresses in channel definitions and client connections to all instances of any »multi-instance« queue manager. The best way to configure network addresses to a multi-instance queue manager depends on the connection:

Queue manager channels

The CONNAME attribute of channels is a comma-separated list of connection names; for example, CONNAME('92.46.1.1(1415), 85.42.3.8(2423)'). The connections are tried in the order specified in the connection list until a connection is successfully established. If no connection is successful, the channel attempts to reconnect.

Cluster channels

Typically, no additional configuration is required to make multi-instance queue managers work in a cluster.

If a queue manager connects to a repository queue manager, the repository discovers the network address of the queue manager from the CONNAME of the CLUSRCVR channel at the queue manager. On TCPIP, the queue manager automatically sets the CONNAME if you omit it, or configure it to blanks. When a standby instance takes over, its IP address replaces the IP address of the previous active instance as the CONNAME.

If it is necessary, you can manually configure CONNAME with the list of network addresses of the queue manager instances.

Client connections

Client connections can use connection lists, or queue manager groups to select alternative connections. Client reconnection to a »multi-instance« queue manager is discussed in Automated client reconnection. Clients need to be compiled to run with WebSphere MQ Version 7.0.1 client libraries or better, and be connected to at least a Version 7.0.1 queue manager.

When failover occurs, reconnection takes some time. The standby queue manager has to complete its startup, and its clients and channels have to detect the connection failure, select the connection to the standby that has newly become active, and reconnect. If the failure takes place during a batch transfer on a message channel, the batch is rolled back and restarted. If the client is in the middle of an MQI call during the reconnection, it must tolerate an extended wait before the call completes.

Switching over is faster that failing over, and takes only as long as stopping one instance of the queue manager and starting another. For a queue manager with only few log records to replay, at best switchover might take of the order of a few seconds. To estimate how long failover takes, you need to add the time that it takes for the WebSphere MQ client, the queue manager, or the file system to detect that the active queue manager or network connection has failed. At best the detection takes of the order of 10 seconds, and might be several minutes, depending on the network and the file system.

Parent topic: »Multi-instance queue managers«

Related concepts
»Automatic client reconnection«

This build: January 26, 2011 11:56:18

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70155_

6.6.3.3.8. Application recovery

Application recovery is the automated continuation of application processing after failover. Application recovery following failover requires careful design. Some applications need to be aware failover has taken place.

The objective of application recovery is for the application to continue processing with only a short delay. Before continuing with new processing, the application must back out and resubmit the unit of work that it was processing during the failure.

A problem for application recovery is losing the context that is shared between the WebSphere® MQ client and the queue manager, and stored in the queue manager. The WebSphere MQ client restores most of the context, but there are some parts of the context that cannot be reliably restored. The following sections describe some properties of application recovery and how they affect the recovery of applications connected to a »multi-instance« queue manager.

Transactional messaging

From the perspective of delivering messages, failover does not change the persistent properties of WebSphere MQ messaging. If messages are persistent, and correctly managed within units of work, then messages are not lost during a failover.

From the perspective of transaction processing, transactions are either backed out or committed after failover.

Uncommitted transactions are rolled back. After failover, a reconnectable application receives an MQRC_BACKED_OUT reason code to indicate that the transaction has failed, and it needs to roll back the transaction by issuing MQBACK. It then needs to restart the transaction again.

Committed transactions are transactions that have reached the second phase of a two-phase commit, or single phase (message only) transactions that have begun MQCMIT.

If the queue manager is the transaction coordinator and MQCMIT has begun the second phase of its two-phase commit before the failure, the transaction successfully completes. The completion is under the control of the queue manager and continues when the queue manager is running again. In a reconnectable application, the MQCMIT call completes normally.

In a single phase commit, which involves only messages, a transaction that has started commit processing completes normally under the control of the queue manager once it is running again. In a reconnectable application, the MQCMIT completes normally.

Reconnectable clients can use single phase transactions under the control of the queue manager as the transaction coordinator. The extended transactional client does not support reconnection. If reconnection is requested when the transactional client connects, the connection succeeds, but without the ability to be reconnected. The connection behaves as if it is not reconnectable.

Application restart or resume

Failover interrupts an application. After a failure an application can restart from the beginning, or it can resume processing following the interruption. The latter is called *automatic client reconnection*. Automatic client reconnect is not supported by WebSphere MQ classes for Java.

With a WebSphere MQ client application, you can set a connection option to reconnect the client automatically. The options are MQCNO_RECONNECT or MQCNO_RECONNECT_Q_MGR. If no option is set, the client does not try to reconnect automatically and the queue manager failure returns MQRC_CONNECTION_BROKEN to the client. You might design the client to try and start a new connection by issuing a new MQCONN or MQCONNX call.

Server programs have to be restarted; they cannot be automatically reconnected by the queue manager at the point they were processing when the queue manager or server failed. WebSphere MQ server programs are typically not restarted on the standby queue manager instance when a »multi-instance« queue manager instance fails.

You can automate a WebSphere MQ server program to restart on the standby server in two ways:

1. Package your server application as a queue manager service. It is restarted when the standby queue manager restarts.
2. Write your own failover logic, triggered for example, by the failover log message written by a standby queue manager instance when it starts. The application instance then needs to call MQCONN or MQCONNX after it starts up to create a connection to the queue manager.

Detecting failover

Some applications do need to be aware of failover, others do not. Consider these two examples.

1. A messaging application that gets or receives messages over a messaging channel does not normally require the queue manager at the other end of the channel to be up and running: it is unlikely to be affected if the queue manager at the other end of the channel restarts on a standby instance.
2. A WebSphere MQ client application processes persistent message input from one queue and puts persistent message responses onto another queue as part of a single unit of work: if it handles an MQRC_BACKED_OUT reason code from MQPUT, MQGET or MQCMIT within sync point by issuing MQBACK and restarting the unit of work, then no messages are lost, and the application does not need to do any special processing to deal with a connection failure.

Suppose however, in the second example, that the application is browsing the queue to select the message to process by using the MQGET option, MQGMO_MSG_UNDER_CURSOR. Reconnection resets the browse cursor, and the MQGET call does not return the correct message. In this example, the application has to be aware failover has occurred, and before issuing another MQGET for the message under the cursor, must restore the browse cursor.

Losing the browse cursor is one example of how the application context changes following reconnection. Other cases are documented in Recovery of an automatically reconnected client.

You have three alternative design patterns for WebSphere MQ client applications following failover. Only one of them does not need to detect the failover.

No reconnection

In this pattern, the application stops all processing on the current connection when the connection is broken. For the application to continue processing, it must establish a new connection with the queue manager. The application is entirely responsible for transferring any state information it requires to continue processing on the new connection. Existing client applications that reconnect with a queue manager after losing their connection are written in this way.

The client receives a reason code, such as MQRC_CONNECTION_BROKEN, or MQRC_Q_MGR_NOT_AVAILABLE from the next MQI call after the connection is lost. The application must discard all its WebSphere MQ state information, such as queue handles, and issue a new MQCONN or MQCONNX call to establish a new connection, and then reopen the WebSphere MQ objects it needs to process.

The default MQI behavior is for the queue manager connection handle to become unusable after a connection with the queue manager is lost. The default is equivalent to setting the MQCNO_RECONNECT_DISABLED option on MQCONNX to prevent application reconnection after failover.

Failover tolerant

Write the application so it is unaffected by failover. Sometimes careful error handling is sufficient to deal with failover.

Reconnection aware

Register an MQCBT_EVENT_HANDLER event handler with the queue manager. The event handler is posted with MQRC_RECONNECTING when the client starts to try to reconnect to the server, and MQRC_RECONNECTED after a successful reconnection. You can then run a routine to reestablish a predictable state so that the client application is able to continue processing.

Recovery of an automatically reconnected client

Failover is an unexpected event, and for an automatically reconnected client to work as designed the consequences of reconnection must be predictable.

A major element of turning an unexpected failure into a predictable and reliable recovery is the use of transactions.

In the previous section, an example, 2, was given of an WebSphere MQ client using a local transaction to coordinate MQGET and MQPUT. The client issues an MQCMIT or MQBACK call in response to a MQRC_BACKED_OUT error and then resubmits the backed out transaction. The queue manager failure causes the transaction to be backed out, and the behavior of the client application ensures no transactions, and no messages, are lost.

Not all program state is managed as part of a transaction, and therefore the consequences of reconnection become harder to understand. You need to know how reconnection changes the state of a WebSphere MQ client in order to design your client application to survive queue manager failover.

You might decide to design your application without any special failover code, handling reconnection errors with the same logic as other errors. Alternatively, you might choose to recognize that reconnection requires special error processing, and register an event handler with WebSphere MQ to run a routine to handle failover. The routine might handle the reconnection processing itself, or set a flag to indicate to the main program thread that when it resumes processing it needs to perform recovery processing.

The WebSphere MQ client environment is aware of failover itself, and restores as much context as it can, following reconnection, by storing some state information in the client, and issuing additional MQI calls on behalf of the client application to restore its WebSphere MQ state. For example, handles to objects that were open at the point of failure are restored, and temporary dynamic queues are opened with the same name. But there are changes that are unavoidable and you need your design to deal with these changes. The changes can be categorized into five kinds:

1. New, or previously undiagnosed errors, are returned from MQI calls until a consistent new context state is restored by the application program.
   An example of receiving a new error is the return code MQRC_CONTEXT_NOT_AVAILABLE when trying to pass context after saving context before the reconnection. The context cannot be restored after reconnection because the security context is not passed to an unauthorized client program. To do so would let a malicious application program obtain the security context.
   Typically, applications handle common and predictable errors in a carefully designed way, and relegate uncommon errors to a generic error handler. The error handler might disconnect from WebSphere MQ and reconnect again, or even stop the program altogether. To improve continuity might need to deal with some errors in a different way.
2. Non-persistent messages might be lost.
3. Transactions are rolled back.
4. MQGET or MQPUT calls used outside a sync point might be interrupted with the possible loss of a message.
5. Timing induced errors, due to a prolonged wait in an MQI call.

Some details about lost context are listed in the following section.

- Non-persistent messages are discarded, unless put to a queue with the NPMCLASS(HIGH) option, and the queue manager failure did not interrupt the option of storing non-persistent messages on shutdown.
- A non-durable subscription is lost when a connection is broken. On reconnection, it is re-established. Consider using a durable subscription.
- The get-wait interval is recomputed; if its limit is exceeded it returns MQRC_NO_MSG_AVAILABLE. Similarly, subscription expiry is recomputed to give the same overall expiry time.
- The position of the browse cursor in a queue is lost; it is typically reestablished before the first message.
  - MQGET calls that specify MQGMO_BROWSE_MSG_UNDER_CURSOR or MQGMO_MSG_UNDER_CURSOR, fail with reason code MQRC_NO_MSG_AVAILABLE.
  - Messages locked for browsing are unlocked.
  - Browse marked messages with handle scope are unmarked and can be browsed again.
  - Cooperatively browse marked messages are unmarked in most cases.
- Security context is lost. Attempts to use saved message context, such as putting a message with MQPMO_PASS_ALL_CONTEXT fail with MQRC_CONTEXT_NOT_AVAILABLE.
- Message tokens are lost. MQGET using a message token returns the reason code MQRC_NO_MSG_AVAILABLE.
  Note: *MsgId* and *CorrelId*, as they are part of the message, are preserved with the message during failover, and so MQGET using MsgId or CorrelId work as expected.
- Messages put on a queue under sync point in an uncommitted transaction are no longer available.
- Processing messages in a logical order, or in a message group, results in a return code of MQRC_RECONNECT_INCOMPATIBLE after reconnection.
- An MQI call might return MQRC_RECONNECT_FAILED rather than the more general MQRC_CONNECTION_BROKEN that clients typically receive today.
- »MQRC_CALL_INTERRUPTED is returned - after a successful reconnect - if the WebSphere MQ client has received no response from the queue manager to indicate the success or failure

of »
- the delivery of a persistent message using an MQPUT call outside sync point.
- the delivery of a persistent message or a message with default persistence using an MQPUT1 call outside sync point.
- the commit of a transaction using an MQCMIT call.

« «

- Channels are restarted as new instances (they might also be different channels), and so no channel exit state is retained.
- Temporary dynamic queues are restored as part of the process of recovering reconnectable clients that had temporary dynamic queues open. No messages on a temporary dynamic queue are restored, but applications that had the queue open, or had remembered the name of the queue, are able to continue processing.
  There is the possibility that if the queue is being used by an application other than the one that created it, that it might not be restored quickly enough to be present when it is next referenced. For example, if a client creates a temporary dynamic queue as a reply-to queue, and a reply message is to be placed on the queue by a channel, the queue might not be recovered in time. In this case, the channel would typically place the reply-to message on the dead letter queue.

Parent topic: »Multi-instance queue managers«

This build: January 26, 2011 11:56:19

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70156_

6.6.3.3.9. Data recovery and high availability

High availability solutions using »multi-instance« queue managers must include a mechanism to recover data after a storage failure.

A »multi-instance« queue manager increases the availability of queue manager processes, but not the availability of other components, such as the file system, that the queue manager uses to store messages, and other information.

»One way to make data highly available is to use networked resilient data storage. You can either build your own solution using a networked file system and resilient data storage, or you can buy an integrated solution. If you want to combine resilience with disaster recovery, then asynchronous disk replication, which permits disk replication over tens, or hundreds of kilometers, is available.«

You can configure the way different WebSphere® MQ directories are mapped to storage media, to make the best use of the media. For »*multi-instance*« queue managers there is an important distinction between two types of WebSphere MQ directories and files.

Directories that must be shared between the instances of a queue manager.

The information that must be shared between different instances of a queue manager is in two directories: the qmgrs and logs directories. The directories must be on a shared networked file system. You are advised to use a storage media that provides continuous high availability and excellent performance because the data is constantly changing as messages are created and deleted.

Directories and files that do not *have* to be shared between instances of a queue manager.

Some other directories do not have to be shared between different instances of a queue manager, and are quickly restored by means other than using a mirrored file system.

- WebSphere MQ executables and the tools directory. Replace by reinstalling or by backing up and restoring from a backed up file archive.
- Configuration information that is modified for the installation as a whole. The configuration information is either managed by WebSphere MQ, such as the mqs.ini file on UNIX systems, or part of your own configuration management such as MQSC configuration scripts. Back up and restore using a file archive.
- Installation-wide output such as traces, error logs and FFDC files. The files are stored in the errors and trace subdirectories in the default data directory. The default data directory on UNIX is /var/mqm . On Windows the default data directory is the WebSphere MQ installation directory.

You can also use a backup queue manager to take regular media backups of a »multi-instance« queue manager using linear logging. A backup queue manager does not provide recovery that is

as fast as from a mirrored file system, and it does not recover changes since the last backup. The backup queue manager mechanism is more appropriate for use in off-site disaster recovery scenarios than recovering a queue manager after a localized storage failure.
Parent topic: »Multi-instance queue managers«

Related concepts
»Backing up and restoring WebSphere MQ«
Using a backup queue manager

This build: January 26, 2011 11:56:19

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70157_

6.6.3.3.10. Combining WebSphere MQ Availability solutions

Applications are using other WebSphere® MQ capabilities to improve availability. »Multi-instance« queue managers complement other high availability capabilities.

WebSphere MQ Clusters increase queue availability

You can increase queue availability by creating multiple definitions of a cluster queue; up to one of every queue on each manager in the cluster.

Suppose a member of the cluster fails and then a new message is sent to a cluster queue. Unless the message *has* to go to the queue manager that has failed, the message is sent to another running queue manager in the cluster that has a definition of the queue.

Although clusters greatly increase availability, there are two related failure scenarios that result in messages getting delayed. Building a cluster with multi-instance queue managers reduces the chance of a message being delayed.

Marooned messages

If a queue manager in the cluster fails, no more messages that can be routed to other queue managers in the cluster are routed to the failed queue manager. Messages that have already been sent are marooned until the failed queue manager is restarted.

Affinities

Affinity is the term used to describe information shared between two otherwise separate computations. For example, an affinity exists between an application sending a request message to a server and the same application expecting to process the reply. Another example would be a sequence of messages, the processing of each message depending on the previous messages.

If you send messages to clustered queues you need to consider affinities. Do you need to send successive messages to the same queue manager, or can each message go to any member of the cluster?

If you do need to send messages to the same queue manager in the cluster and it fails, the messages wait in the transmission queue of the sender until the failed cluster queue manager is running again.

If the cluster is configured with multi-instance queue managers the delay waiting for the failed queue manager to restart is limited to the order of a minute or so while the standby takes over. When the standby is running, marooned messages resume processing, channels to the newly activated queue manager instance are started, and the messages that were waiting in transmission queues start flowing.

A possible way to configure a cluster to overcome messages being delayed by a failed queue manager, is to deploy two different queue managers to each server in the cluster, and arrange for one to be the active and one to be the standby instance of the different queue managers. This is an active-standby configuration, and it increases the availability of the cluster.

As well as having the benefits of reduced administration and increased scalability, clusters continue to provide additional elements of availability to complement »multi-instance« queue managers. Clusters protect against other types of failure that affect both the active and standby instances of a queue manager.

Uninterrupted service

A cluster provides an uninterrupted service. New messages received by the cluster are sent to active queue managers to be processed. Do not rely on a »multi-instance« queue manager to provide an uninterrupted service because it takes time for the standby queue manager to

detect the failure and complete its startup, for its channels to be reconnected, and for failed batches of messages to be resubmitted.

Localized outage

There are practical limitations to how far apart the active, standby, and file system servers can be, as they need to interact at millisecond speeds to deliver acceptable performance.
Clustered queue managers require interaction speeds of the order of many seconds, and can be geographically dispersed anywhere in the world.

Operational error

By using two different mechanisms to increase availability you reduce the chances that an operational error, such as a human error, compromises your availability efforts.

Queue sharing groups increase message processing availability

Queue sharing groups, provided only on z/OS®, allow a group of queue managers to share servicing a queue. If one queue manager fails, the other queue managers continue to process all the messages on the queue. »Multi-instance« queue managers are not supported on z/OS and complement queue sharing groups only as part of a wider messaging architecture.

WebSphere MQ Clients increase application availability

WebSphere MQ client programs can connect to different queue managers in a queue manager group based on queue manager availability, connection weightings, and affinities. By running an application on a different machine from the one on which the queue manager is running, you can to improve the overall availability of a solution as long as there is a way to reconnect the application if the queue manager instance it is connected to fails.

Queue manager groups are used to increase client availability by uncoupling a client from a queue manager that is stopped, and load balancing client connections across a group of queue managers, rather like an IP sprayer. The client application must have no affinities with the failed queue manager, such as a dependency on a particular queue, or it cannot resume processing.

Automatic client reconnection and multi-instance queue managers increase client availability by resolving some affinity problems. Automatic client reconnect is not supported by WebSphere MQ classes for Java.

You can set the MQCNO option MQCNO_RECONNECT_Q_MGR, to force a client to reconnect to the same queue manager:

1. If the previously connected single instance queue manager is not running the connection is tried again until the queue manager is running again.
2. If the queue manager is configured as a multi-instance queue manager, then the client reconnects to whichever instance is active.

By automatically reconnecting to the same queue manager, much of the state information the queue manager was holding on behalf of the client, such as the queues it had open and the topic it was subscribed to, are restored. If the client had opened a dynamic reply-to queue to receive a reply to a request, the connection to the reply-to queue is restored too.

Parent topic: »Multi-instance queue managers«

Related information
Reviewing applications for message affinities
handling message affinities
More than one instance of a cluster queue
Queue manager groups

This build: January 26, 2011 11:56:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa70158_

6.6.4. Making sure that messages are not lost (logging)

»WebSphere® MQ logs all the information you need to recover from a queue manager failure.«

WebSphere MQ records all significant changes to the data controlled by the queue manager in a recovery log.

This includes creating and deleting objects, persistent message updates, transaction states, changes to object attributes, and channel activities. The log contains the information you need to recover all updates to message queues by:

- Keeping records of queue manager changes
- Keeping records of queue updates for use by the restart process
- Enabling you to restore data after a hardware or software failure

However, WebSphere MQ also relies on the disk system hosting its files. If the disk system is itself unreliable, information, including log information, can still be lost.

What logs look like
Logs consist of primary and secondary files, and a control file. You define the number and size of log files and where they are stored in the file system.

Types of logging
Circular and linear logging.

Using checkpointing to ensure complete recovery
Checkpoints synchronize queue manager data and log files, and mark a point of consistency from which log records can be discarded. Frequent checkpointing makes recovery quicker.

Calculating the size of the log
Estimating the size of log a queue manager needs.

Managing logs
Logs are nearly self-managing, but sometimes need managing to resolve space problems.

Using the log for recovery
Using logs to recover from failures.

Protecting WebSphere MQ log files
Do not touch the log files when a queue manager is running, recovery might be impossible. Use super user or mqm authority to protect log files against inadvertent modification.

Dumping the contents of the log using the dmpmqlog command
How to use the dmpmqlog command to dump the contents of the queue manager log.

Parent topic: Availability, recovery and restart

This build: January 26, 2011 11:55:13

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14480_

6.6.4.1. What logs look like

Logs consist of primary and secondary files, and a control file. You define the number and size of log files and where they are stored in the file system.

A WebSphere® MQ log consists of two components:

1. One or more files of log data.
2. A log control file

A file of log data is also known as a log extent.

There are a number of log files that contain the data being recorded. You can define the number and size (as explained in Configuring WebSphere MQ), or take the system default of three files.

In WebSphere MQ for Windows, each of the three files defaults to 1 MB. In WebSphere MQ for UNIX systems, each of the three files defaults to 4 MB.

When you create a queue manager, the number of log files you define is the number of *primary* log files allocated. If you do not specify a number, the default value is used.

In WebSphere MQ for Windows, if you have not changed the log path, log files are created under the directory:

```
C:\Program Files\IBM\WebSphere MQ\log\<QMgrName>
```

In WebSphere MQ for UNIX systems, if you have not changed the log path, log files are created under the directory:

```
/var/mqm/log/<QMgrName>
```

WebSphere MQ starts with these primary log files, but if the primary log space is not sufficient, it allocates *secondary* log files. It does this dynamically and removes them when the demand for log space reduces. By default, up to two secondary log files can be allocated. You can change this default allocation, as described in Configuring WebSphere MQ.

The log control file

Parent topic: Making sure that messages are not lost (logging)

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14490_

6.6.4.1.1. The log control file

The log control file contains the information needed to control the use of log files, such as their size and location, the name of the next available file, and so on.

Note: Ensure that the logs created when you start a queue manager are large enough to accommodate the size and volume of messages that your applications will handle. You will probably need to change the default log numbers and sizes to meet your requirements. For more information, see Calculating the size of the log.

Parent topic: What logs look like

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14500_

6.6.4.2. Types of logging

Circular and linear logging.

In WebSphere® MQ, the number of files that are required for logging depends on the file size, the number of messages you have received, and the length of the messages. There are two ways of maintaining records of queue manager activities: circular logging and linear logging.

Circular logging

Linear logging

Parent topic: Making sure that messages are not lost (logging)

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14510_

6.6.4.2.1. Circular logging

Use circular logging if all you want is restart recovery, using the log to roll back transactions that were in progress when the system stopped.

Circular logging keeps all restart data in a ring of log files. Logging fills the first file in the ring, then moves on to the next, and so on, until all the files are full. It then

goes back to the first file in the ring and starts again. This continues as long as the product is in use, and has the advantage that you never run out of log files.

WebSphere® MQ keeps the log entries required to restart the queue manager without loss of data until they are no longer required to ensure queue manager data recovery. The mechanism for releasing log files for reuse is described in Using checkpointing to ensure complete recovery.

Parent topic: Types of logging

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14520_

6.6.4.2.2. Linear logging

Use linear logging if you want both restart recovery and media recovery (recreating lost or damaged data by replaying the contents of the log).

Linear logging keeps the log data in a continuous sequence of files. Space is not reused, so you can always retrieve any record logged in any log extent that has not been deleted

As disk space is finite, you might have to think about some form of archiving. It is an administrative task to manage your disk space for the log, reusing or extending the existing space as necessary.

The number of log files used with linear logging can be very large, depending on your message flow and the age of your queue manager. However, there are a number of files that are said to be *active*. Active files contain the log entries required to restart the queue manager. Collectively, active log files are known as the *active log*. The number of active log files is usually less than the number of primary log files as defined in the configuration files. (See Calculating the size of the log for information about defining the number.)

The key event that controls whether a log file is termed active or not is a *checkpoint*. A WebSphere® MQ checkpoint is a point of consistency between the recovery log and object files. A checkpoint determines the set of log files needed to perform restart recovery. Log files that are not active are not required for restart recovery, and are termed inactive. In some cases inactive log files are required for media recovery. (See Using checkpointing to ensure complete recovery for further information about checkpointing.)

Inactive log files can be archived as they are not required for restart recovery. Inactive log files that are not required for media recovery can be considered as superfluous log files. You can delete superfluous log files if they are no longer of interest to your operation. Refer to Managing logs for further information about the disposition of log files.

If a new checkpoint is recorded in the second, or later, primary log file, the first file can become inactive and a new primary file is formatted and added to the end of the primary pool, restoring the number of primary files available for logging. In this way the primary log file pool can be seen to be a current set of files in an ever-extending list of log files. Again, it is an administrative task to manage the inactive files according to the requirements of your operation.

Although secondary log files are defined for linear logging, they are not used in normal operation. If a situation arises when, probably due to long-lived transactions, it is not possible to free a file from the active pool because it might still be required for a restart, secondary files are formatted and added to the active log file pool.

If the number of secondary files available is used up, requests for most further operations requiring log activity will be refused with an MQRC_RESOURCE_PROBLEM return code being returned to the application.

Both types of logging can cope with unexpected loss of power, assuming that there is no hardware failure.

Parent topic: Types of logging

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa14530_

6.6.4.3. Using checkpointing to ensure complete recovery

Checkpoints synchronize queue manager data and log files, and mark a point of consistency from which log records can be discarded. Frequent checkpointing makes recovery quicker.

Persistent updates to message queues happen in two stages. First, the records representing the update are written to the log, then the queue file is updated. The log files can thus become more up-to-date than the queue files. To ensure that restart processing begins from a consistent point, WebSphere® MQ uses checkpoints. A checkpoint is a point in time when the record described in the log is the same as the record in the queue. The checkpoint itself consists of the series of log records needed to restart the queue manager; for example, the state of all transactions (units of work) active at the time of the checkpoint.

WebSphere MQ generates checkpoints automatically. They are taken when the queue manager starts, at shutdown, when logging space is running low, and after every 10 000 operations logged.

As the queues handle further messages, the checkpoint record becomes inconsistent with the current state of the queues.

When WebSphere MQ restarts, it finds the latest checkpoint record in the log. This information is held in the checkpoint file that is updated at the end of every checkpoint. The checkpoint record represents the most recent point of consistency between the log and the data. All the operations that have taken place since the checkpoint are replayed forward. This is known as the replay phase. The replay phase brings the queues back to the logical state they were in before the system failure or shutdown. During the replay phase a list is created of the transactions that were in-flight when the system failure or shutdown occurred. Messages AMQ7229 and AMQ7230 are issued to indicate the progression of the replay phase.

In order to know which operations to back out or commit, WebSphere MQ accesses each active log record associated with an in-flight transaction. This is known as the recovery phase. Messages AMQ7231, AMQ7232 and AMQ7234 are issued to indicate the progression of the recovery phase.

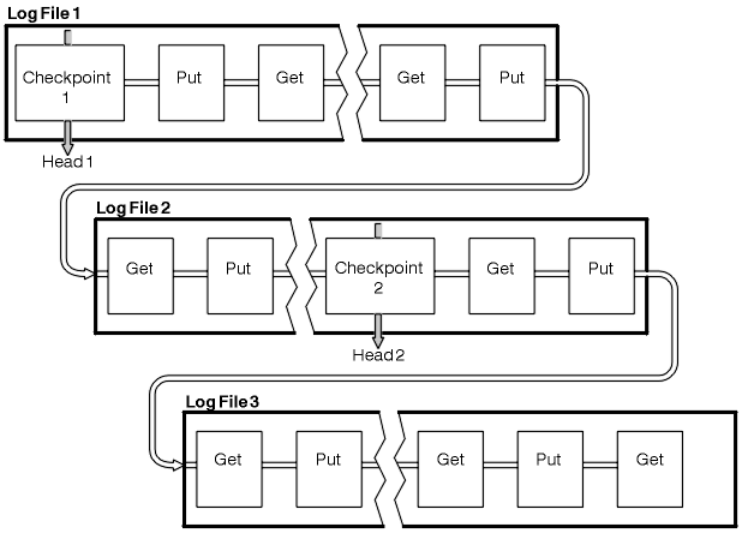
Once all the necessary log records have been accessed during the recovery phase, each active transaction is in turn resolved and each operation associated with the transaction will be either backed out or committed. This is known as the resolution phase. Message AMQ7233 is issued to indicate the progression of the resolution phase.

WebSphere MQ maintains internal pointers to the head and tail of the log. It moves the head pointer to the most recent checkpoint consistent with recovering message data.

Checkpoints are used to make recovery more efficient, and to control the reuse of primary and secondary log files.

In Figure 1, all records before the latest checkpoint, Checkpoint 2, are no longer needed by WebSphere MQ. The queues can be recovered from the checkpoint information and any later log entries. For circular logging, any freed files before the checkpoint can be reused. For a linear log, the freed log files no longer need to be accessed for normal operation and become inactive. In the example, the queue head pointer is moved to point at the latest checkpoint, Checkpoint 2, which then becomes the new queue head, Head 2. Log File 1 can now be reused.

Figure 1. Checkpointing. For simplicity, only the ends of the log files are shown.



»Checkpointing with long-running transactions«

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14540_

6.6.4.3.1. Checkpointing with long-running transactions

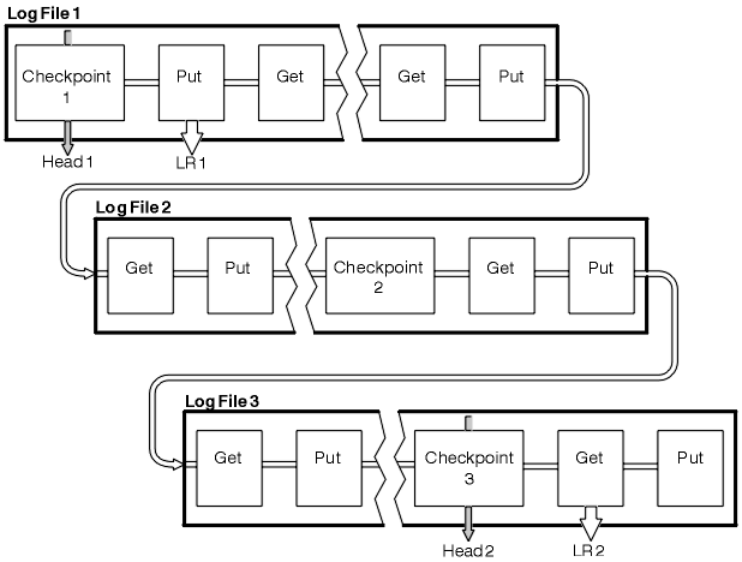
Figure 1 shows how a long-running transaction affects reuse of log files. In the example, a long-running transaction has made an entry to the log, shown as LR 1, after the first checkpoint shown. The transaction does not complete (at point LR 2) until after the third checkpoint. All the log information from LR 1 onwards is retained to allow recovery of that transaction, if necessary, until it has completed.

After the long-running transaction has completed, at LR 2, the head of the log is moved to Checkpoint 3, the latest logged checkpoint. The files containing log records before Checkpoint 3, Head 2, are no longer needed. If you are using circular logging, the space can be reused.

If the primary log files are completely full before the long-running transaction completes, secondary log files are used to avoid the logs getting full.

When the log head is moved and you are using circular logging, the primary log files might become eligible for reuse and the logger, after filling the current file, reuses the first primary file available to it. If you are using linear logging, the log head is still moved down the active pool and the first file becomes inactive. A new primary file is formatted and added to the bottom of the pool in readiness for future logging activities.

Figure 1. Checkpointing with a long-running transaction. For simplicity, only the ends of the log files are shown.



Parent topic: »Using checkpointing to ensure complete recovery«

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14550_

6.6.4.4. Calculating the size of the log

»Estimating the size of log a queue manager needs.«

After deciding whether the queue manager should use circular or linear logging, you need to estimate the size of the log that the queue manager needs. The size of the log is determined by the following log configuration parameters:

LogFilePages

The size of each primary and secondary log file in units of 4K pages

LogPrimaryFiles

The number of preallocated primary log files

LogSecondaryFiles

The number of secondary log files that can be created for use when the primary log files are full

Table 1 shows the amount of data the queue manager logs for various operations. Most queue manager operations need a minimal amount of log space. However, when a persistent message is put to a queue, all the message data must be written to the log to make it possible to recover the message. The size of the log depends, typically, on the number and size of the persistent messages the queue manager needs to handle.

Table 1. Log entry sizes (all values are approximate)

Operation	Size
Put persistent message	750 bytes + message length If the message is large, it is divided into segments of 15700 bytes, each segment adding an extra 300-bytes.
Get message	260 bytes
Sync point, commit	750 bytes
Sync point, rollback	1000 bytes + 12 bytes for each get or put to be rolled back
Create object	1500 bytes
Delete object	300 bytes
Alter attributes	1024 bytes
Record media image	800 bytes + image The image is divided into segments of 260 000 bytes, each segment adding an extra 300-bytes.

Checkpoint	750 bytes + 200 bytes for each active unit of work Additional data might be logged for any uncommitted puts or gets that have been buffered for performance reasons.

Note:

1. You can change the number of primary and secondary log files each time the queue manager starts.
2. You cannot change the log file size; you must determine it before creating the queue manager.
3. The number of primary log files and the log file size determine the amount of log space that is preallocated when the queue manager is created.
4. The total number of primary and secondary log files cannot exceed 511 on UNIX systems, or 255 on Windows, which in the presence of long-running transactions, limits the maximum amount of log space available to the queue manager for restart recovery. The amount of log space the queue manager might need for media recovery does not share this limit.
5. When *circular* logging is being used, the queue manager reuses primary log space. This means that the queue manager's log can be smaller than the amount of data you have estimated that the queue manager needs to log. The queue manager will, up to a limit, allocate a secondary log file when a log file becomes full, and the next primary log file in the sequence is not available.
6. Primary log files are made available for reuse during a checkpoint. The queue manager takes both the primary and secondary log space into consideration before taking a checkpoint because the amount of log space is running low.
   If you do not define more primary log files than secondary log files, the queue manager might allocate secondary log files before a checkpoint is taken. This makes the primary log files available for reuse.

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:13

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14560_

6.6.4.5. Managing logs

»Logs are nearly self-managing, but sometimes need managing to resolve space problems.«

Over time, some of the log records written become unnecessary for restarting the queue manager. If you are using circular logging, the queue manager reclaims freed space in the log files. This activity is not apparent to the user and you do not usually see the amount of disk space used reduce because the space allocated is quickly reused.

Of the log records, only those written since the start of the last complete checkpoint, and those written by any active transactions, are needed to restart the queue manager. Thus, the log might fill if a checkpoint has not been taken for a long time, or if a long-running transaction wrote a log record a long time ago. The queue manager tries to take checkpoints often enough to avoid the first problem.

When a long-running transaction fills the log, attempts to write log records fail and some MQI calls return MQRC_RESOURCE_PROBLEM. (Space is reserved to commit or roll back all in-flight transactions, so MQCMIT or MQBACK should not fail.)

The queue manager rolls back transactions that consume too much log space. An application whose transaction is rolled back in this way cannot perform subsequent MQPUT or MQGET operations specifying sync point under the same transaction. An attempt to put or get a message under sync point in this state returns MQRC_BACKED_OUT. The application can then issue MQCMIT, which returns MQRC_BACKED_OUT, or MQBACK and start a new transaction. When the transaction consuming too much log space has been rolled back, its log space is released and the queue manager continues to operate normally.

If the log fills, message AMQ7463 is issued. In addition, if the log fills because a long-running transaction has prevented the space being released, message AMQ7465 is issued.

Finally, if records are being written to the log faster than the asynchronous housekeeping processes can handle them, message AMQ7466 is issued. If you see this message, increase the number of log files or reduce the amount of data being processed by the queue manager.

»What happens when a disk gets full«

»Managing log files«
Allocate sufficient space for your log files. For linear logging, you can delete old log files when they are no longer required.

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14570_

6.6.4.5.1. What happens when a disk gets full

The queue manager logging component can cope with a full disk, and with full log files. If the disk containing the log fills, the queue manager issues message AMQ6708 and an error record is taken.

The log files are created at their maximum size, rather than being extended as log records are written to them. This means that WebSphere® MQ can run out of disk space only when it is creating a new file; it cannot run out of space when it is writing a record to the log. WebSphere MQ always knows how much space is available in the existing log files, and manages the space within the files accordingly.

If you fill the drive containing the log files, you might be able to free some disk space. If you are using a linear log, there might be some inactive log files in the log directory, and you can copy these files to another drive or device. If you still run out of space, check that the configuration of the log in the queue manager configuration file is correct. You might be able to reduce the number of primary or secondary log files so that the log does not outgrow the available space. You cannot alter the size of the log files for an existing queue manager. The queue manager assumes that all log files are the same size.

Parent topic: »Managing logs«

This build: January 26, 2011 11:55:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14580_

6.6.4.5.2. Managing log files

Allocate sufficient space for your log files. For linear logging, you can delete old log files when they are no longer required.

If you are using circular logging, ensure that there is sufficient space to hold the log files when you configure your system (see Log defaults for WebSphere MQ and Queue manager logs). The amount of disk space used by the log does not increase beyond the configured size, including space for secondary files to be created when required.

If you are using a linear log, the log files are added continually as data is logged, and the amount of disk space used increases with time. If the rate of data being logged is high, disk space is consumed rapidly by new log files.

Over time, the older log files for a linear log are no longer needed to restart the queue manager or to perform media recovery of any damaged objects. The following are methods for determining which log files are still required:

Logger event messages

When enabled, logger event messages are generated when queue managers starts writing log records to a new log file. The contents of logger event messages specify the log files that are still required for queue manager restart, and media recovery. For more information on logger event messages, see Monitoring WebSphere MQ

Queue manager status

Executing the MQSC command, DISPLAY QMSTATUS, or the PCF command, Inquire Queue Manager Status, returns queue manager information, including details of the required log files. For more information on MQSC commands, see the WebSphere MQ Script (MQSC) Command Reference manual, and for information on PCF commands, see the WebSphere MQ Programmable Command

Formats and Administration Interface manual.

Queue manager messages

Periodically, the queue manager issues a pair of messages to indicate which of the log files are needed:

- Message AMQ7467 gives the name of the oldest log file needed to restart the queue manager. This log file and all newer log files must be available during queue manager restart.
- Message AMQ7468 gives the name of the oldest log file needed for media recovery.

Only log files required for queue manager restart, active log files, need to be online. Inactive log files can be copied to an archive medium such as tape for disaster recovery, and removed from the log directory. Inactive log files that are not required for media recovery can be considered as superfluous log files. You can delete superfluous log files if they are no longer of interest to your operation.

To determine "older" and "newer" log files, use the log file number rather than the modification times applied by the file system.

If any log file that is needed cannot be found, operator message AMQ6767 is issued. Make the log file, and all subsequent log files, available to the queue manager and retry the operation.

Note: »When performing media recovery, all the required log files must be available in the log file directory at the same time. Make sure that you take regular media images of any objects you might wish to recover to avoid running out of disk space to hold all the required log files. To take a media image of all your objects in your queue manager, run the rcdmqimg command as shown in the following examples:

On Windows

```
rcdmqimg -m QMNAME -t all *
```

On Unix

```
rcdmqimg -m QMNAME -t all "*"
```

Running rcdmqimg moves the media log sequence number (LSN) forwards. For further details on log sequence numbers, see Dumping the contents of the log using the dmpmqlog command. rcdmqimg does not run automatically, therefore must be run manually or from an automatic task you have created. For more information about this command, see rcdmqimg and dmpmqlog.«

Note: »Messages AMQ7467 and AMQ7468 can also be issued at the time of running the rcdmqimg command.«

»Determining superfluous log files«
When managing linear log files, it is important to be sure which files can be deleted or archived. This information will assist you in making this decision.

»Log file location«

Parent topic: »Managing logs«

This build: January 26, 2011 11:55:14

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14590_

6.6.4.5.2.1. Determining superfluous log files

When managing linear log files, it is important to be sure which files can be deleted or archived. This information will assist you in making this decision.

Do not use the file system's modification times to determine "older" log files. Use only the log file number. The queue manager's use of log files follows complex rules, including pre-allocation and formatting of log files before they are needed. You might see log files with modification times that would be misleading if you try to use these times to determine relative age.

To determine the oldest log file needed to restart the queue manager, issue the command DISPLAY QMSTATUS RECLOG.

To determine the oldest log file needed to perform media recovery, issue the command DISPLAY QMSTATUS MEDIALOG.

In general a lower log file number implies an older log. Unless you have a very high log file turnover, of the order of 3000 log files per day for 10 years, then you do not need to cater for the number wrapping at 9 999 999. In this case, you can archive any log file with a number less than the RECLOG value, and you can delete any log file with a number less than both the RECLOG and MEDIALOG values.

If however you have a very high turnover of log files, or otherwise want to be confident of coping with the general case, then the following algorithm can typically be used:

```
Let S == restart log file number ( from DISPLAY QMSTATUS RECLOG).
Let M == media recovery log file number (from DISPLAY QMSTATUS MEDIALOG).
Let L == a given log file number whose eligibility for deletion or archiving needs to be
determined.

function minlog (a, b) {
     if (abs (a - b) < 5000000)
          return min (a, b);   # Not wrapped.
     else
          return max (a, b);   # Wrapped. }

A log file L can be deleted if (L != S && L != M && minlog (L, minlog (S, M)) == L).
A log file L can be archived if (L != S && minlog (L, S) == L).
```

Parent topic: »Managing log files«

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23320_

6.6.4.5.2.2. Log file location

When choosing a location for your log files, remember that operation is severely impacted if WebSphere® MQ fails to format a new log because of lack of disk space.

If you are using a circular log, ensure that there is sufficient space on the drive for at least the configured primary log files. Also leave space for at least one secondary log file, which is needed if the log has to grow.

If you are using a linear log, allow considerably more space; the space consumed by the log increases continuously as data is logged.

Ideally, place the log files on a separate disk drive from the queue manager data. This has benefits in terms of performance. It might also be possible to place the log files on multiple disk drives in a mirrored arrangement. This protects against failure of the drive containing the log. Without mirroring, you could be forced to go back to the last backup of your WebSphere MQ system.

Parent topic: »Managing log files«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14600_

6.6.4.6. Using the log for recovery

»Using logs to recover from failures.«

There are several ways that your data can be damaged. WebSphere® MQ helps you to recover from:

- A damaged data object
- A power loss in the system
- A communications failure

This section looks at how the logs are used to recover from these problems.

»Recovering from power loss or communications failures«

»Recovering damaged objects«

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14610_

6.6.4.6.1. Recovering from power loss or communications failures

WebSphere® MQ can recover from both communications failures and loss of power. In addition, it can sometimes recover from other types of problem, such as inadvertent deletion of a file.

In the case of a communications failure, messages remain on queues until they are removed by a receiving application. If the message is being transmitted, it remains on the transmission queue until it can be successfully transmitted. To recover from a communications failure, you can usually restart the channels using the link that failed.

If you lose power, when the queue manager is restarted WebSphere MQ restores the queues to their committed state at the time of the failure. This ensures that no persistent messages are lost. Nonpersistent messages are discarded; they do not survive when WebSphere MQ stops abruptly.

Parent topic: »Using the log for recovery«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14620_

6.6.4.6.2. Recovering damaged objects

There are ways in which a WebSphere® MQ object can become unusable, for example because of inadvertent damage. You then have to recover either your complete system or some part of it. The action required depends on when the damage is detected, whether the log method selected supports media recovery, and which objects are damaged.

»Media recovery«
Media recovery re-creates objects from information recorded in a linear log. For example, if an object file is inadvertently deleted, or becomes unusable for some other reason, media recovery can re-create it. The information in the log required for media recovery of an object is called a media image.

»Recovering from media images«

»Recovering damaged objects during start up«

»Recovering damaged objects at other times«

Parent topic: »Using the log for recovery«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14630_

6.6.4.6.2.1. Media recovery

»Media recovery re-creates objects from information recorded in a linear log. For example, if

an object file is inadvertently deleted, or becomes unusable for some other reason, media recovery can re-create it. The information in the log required for media recovery of an object is called a media image.

A media image is a sequence of log records containing an image of an object from which the object itself can be re-created.

The first log record required to re-create an object is known as its *media recovery record*; it is the start of the latest media image for the object. The media recovery record of each object is one of the pieces of information recorded during a checkpoint.

When an object is re-created from its media image, it is also necessary to replay any log records describing updates performed on the object since the last image was taken.

Consider, for example, a local queue that has an image of the queue object taken before a persistent message is put onto the queue. In order to re-create the latest image of the object, it is necessary to replay the log entries recording the putting of the message to the queue, as well as replaying the image itself.

When an object is created, the log records written contain enough information to completely re-create the object. These records make up the object's first media image. Subsequently, at each shutdown, the queue manager records media images automatically as follows:

- Images of all process objects and queues that are not local
- Images of empty local queues

Media images can also be recorded manually using the rcdmqimg command, described in rcdmqimg. This command writes a media image of the WebSphere® MQ object. Once this has been done, only the logs that hold the media image, and all the logs created after this time, are needed to re-create damaged objects. The benefit of doing this depends on such factors as the amount of free storage available, and the speed at which log files are created.

Parent topic: Recovering damaged objects

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14640_

6.6.4.6.2.2. Recovering from media images

WebSphere® MQ automatically recovers some objects from their media image if it finds that they are corrupt or damaged. In particular, this applies to objects found to be damaged during the normal queue manager startup. If any transaction was incomplete when the queue manager last shutdown, any queue affected is also recovered automatically in order to complete the startup operation.

You must recover other objects manually, using the rcrmqobj command, which replays the records in the log to re-create the WebSphere MQ object. The object is re-created from its latest image found in the log, together with all applicable log events between the time the image was saved and the time the re-create command was issued. If a WebSphere MQ object becomes damaged, the only valid actions that can be performed are either to delete it or to re-create it by this method. Nonpersistent messages cannot be recovered in this way.

See rcrmqobj for further details of the rcrmqobj command.

The log file containing the media recovery record, and all subsequent log files, must be available in the log file directory when attempting media recovery of an object. If a required file cannot be found, operator message AMQ6767 is issued and the media recovery operation fails. If you do not take regular media images of the objects that you want to re-create, you might have insufficient disk space to hold all the log files required to re-create an object.

Parent topic: Recovering damaged objects

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14650_

6.6.4.6.2.3. Recovering damaged objects during start up

If the queue manager discovers a damaged object during startup, the action it takes depends on the type of object and whether the queue manager is configured to support media recovery.

If the queue manager object is damaged, the queue manager cannot start unless it can recover the object. If the queue manager is configured with a linear log, and thus supports media recovery, WebSphere® MQ automatically tries to re-create the queue manager object from its media images. If the log method selected does not support media recovery, you can either restore a backup of the queue manager or delete the queue manager.

If any transactions were active when the queue manager stopped, the local queues containing the persistent, uncommitted messages put or got inside these transactions are also needed to start the queue manager successfully. If any of these local queues is found to be damaged, and the queue manager supports media recovery, it automatically tries to re-create them from their media images. If any of the queues cannot be recovered, WebSphere MQ cannot start.

If any damaged local queues containing uncommitted messages are discovered during startup processing on a queue manager that does not support media recovery, the queues are marked as damaged objects and the uncommitted messages on them are ignored. This is because it is not possible to perform media recovery of damaged objects on such a queue manager and the only action left is to delete them. Message AMQ7472 is issued to report any damage.

Parent topic: »Recovering damaged objects«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14660_

6.6.4.6.2.4. Recovering damaged objects at other times

Media recovery of objects is automatic only during startup. At other times, when object damage is detected, operator message AMQ7472 is issued and most operations using the object fail. If the queue manager object is damaged at any time after the queue manager has started, the queue manager performs a preemptive shutdown. When an object has been damaged you can delete it or, if the queue manager is using a linear log, attempt to recover it from its media image using the rcrmqobj command (see rcrmqobj for further details).

Parent topic: »Recovering damaged objects«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14670_

6.6.4.7. Protecting WebSphere MQ log files

»Do not touch the log files when a queue manager is running, recovery might be impossible. Use super user or mqm authority to protect log files against inadvertent modification.«

Do not remove the active log files manually when a WebSphere® MQ queue manager is running. If a user inadvertently deletes the log files that a queue manager needs to restart, WebSphere MQ does not issue any errors and continues to process data *including persistent messages*. The queue manager shuts down normally, but can fail to restart. Recovery of messages then becomes impossible.

Users with the authority to remove logs that are being used by an active queue manager also have authority to delete other important queue manager resources (such as queue files, the object catalog, and WebSphere MQ executables). They can therefore damage, perhaps through inexperience, a running or dormant queue manager in a way against which WebSphere MQ cannot protect itself.

Exercise caution when conferring super user or mqm authority.

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa14680_

6.6.4.8. Dumping the contents of the log using the dmpmqlog command

How to use the dmpmqlog command to dump the contents of the queue manager log.

Use the dmpmqlog command to dump the contents of the queue manager log. By default all active log records are dumped, that is, the command starts dumping from the head of the log (usually the start of the last completed checkpoint).

The log can usually be dumped only when the queue manager is not running. Because the queue manager takes a checkpoint during shutdown, the active portion of the log usually contains a small number of log records. However, you can use the dmpmqlog command to dump more log records using one of the following options to change the start position of the dump:

- **Start dumping from the *base* of the log. The base of the log is the first log record in the log file that contains the head of the log. The amount of additional data dumped in this case depends on where the head of the log is positioned in the log file. If it is near the start of the log file, only a small amount of additional data is dumped. If the head is near the end of the log file, significantly more data is dumped.**
- **Specify the start position of the dump as an individual log record. Each log record is identified by a unique *log sequence number (LSN)*. In the case of circular logging, this starting log record cannot be before the base of the log; this restriction does not apply to linear logs. You might need to reinstate inactive log files before running the command. You must specify a valid LSN, taken from previous dmpmqlog output, as the start position.**
  For example, with linear logging you can specify the nextlsn from your last dmpmqlog output. The nextlsn appears in Log File Header and indicates the LSN of the next log record to be written. Use this as a start position to format all log records written since the last time the log was dumped.
- **For linear logs only, you can instruct dmpmqlog to start formatting log records from any given log file extent. In this case, dmpmqlog expects to find this log file, and each successive one, in the same directory as the active log files. This option does not apply to circular logs, where dmpmqlog cannot access log records prior to the base of the log.**

The output from the dmpmqlog command is the Log File Header and a series of formatted log records. The queue manager uses several log records to record changes to its data.

Some of the information that is formatted is only of use internally. The following list includes the most useful log records:

Log File Header

Each log has a single log file header, which is always the first thing formatted by the dmpmqlog command. It contains the following fields:

logactive	The number of primary log extents.
loginactive	The number of secondary log extents.
logsize	The number of 4 KB pages per extent.
baselsn	The first LSN in the log extent containing the head of the log.
nextlsn	The LSN of the next log record to be written.
headlsn	The LSN of the log record at the head of the log.
tailsn	The LSN identifying the tail position of the log.
hflag1	Whether the log is CIRCULAR or LOG RETAIN (linear).
HeadExtentID	The log extent containing the head of the log.

Log Record Header

Each log record within the log has a fixed header containing the following information:

LSN	The log sequence number.
LogRecdType	The type of the log record.
XTranid	The transaction identifier associated with this log record (if any). A *TranType* of MQI indicates a WebSphere® MQ-only transaction. A *TranType* of XA is involved with other resource managers. Updates involved within the same unit of work have the same *XTranid*.
QueueName	The queue associated with this log record (if any).

Qid	The unique internal identifier for the queue.
PrevLSN	The LSN of the previous log record within the same transaction (if any).

Start Queue Manager

This logs that the queue manager has started.

StartDate	The date that the queue manager started.
StartTime	The time that the queue manager started.

Stop Queue Manager

This logs that the queue manager has stopped.

StopDate	The date that the queue manager stopped.
StopTime	The time that the queue manager stopped.
ForceFlag	The type of shutdown used.

Start Checkpoint

This denotes the start of a queue manager checkpoint.

End Checkpoint

This denotes the end of a queue manager checkpoint.

ChkPtLSN	The LSN of the log record that started this checkpoint.

Put Message

This logs a persistent message put to a queue. If the message was put under sync point, the log record header contains a non-null *XTranid*. The remainder of the record contains:

SpcIndex	An identifier for the message on the queue. It can be used to match the corresponding **MQGET** that was used to get this message from the queue. In this case a subsequent *Get Message* log record can be found containing the same QueueName and SpcIndex. At this point the SpcIndex identifier can be reused for a subsequent put message to that queue.
Data	Contained in the hex dump for this log record is various internal data followed by the Message Descriptor (eyecatcher MD) and the message data itself.

Put Part

Persistent messages that are too large for a single log record are logged as a single *Put Message* record followed by multiple *Put Part* log records.

Data	Continues the message data where the previous log record left off.

Get Message

Only gets of persistent messages are logged. If the message was got under sync point, the log record header contains a non-null *XTranid*. The remainder of the record contains:

SpcIndex	Identifies the message that was retrieved from the queue. The most recent *Put Message* log record containing the same *QueueName* and *SpcIndex* identifies the message that was retrieved.
QPriority	The priority of the message retrieved from the queue.

Start Transaction

Indicates the start of a new transaction. A TranType of MQI indicates a WebSphere MQ-only transaction. A TranType of XA indicates one that involves other resource managers. All updates made by this transaction will have the same *XTranid*.

Prepare Transaction

Indicates that the queue manager is prepared to commit the updates associated with the specified *XTranid*. This log record is written as part of a two-phase commit involving other resource managers.

Commit Transaction

Indicates that the queue manager has committed all updates made by a transaction.

Rollback Transaction

This denotes the queue manager's intention to roll back a transaction.

End Transaction

This denotes the end of a rolled-back transaction.

Transaction Table

This record is written during sync point. It records the state of each transaction that has made persistent updates. For each transaction the following information is recorded:

XTranid	The transaction identifier.
FirstLSN	The LSN of the first log record associated with the transaction.
LastLSN	The LSN of the last log record associated with the transaction.

Transaction Participants

This log record is written by the XA Transaction Manager component of the queue manager. It records the external resource managers that are participating in transactions. For each participant the following is recorded:

RMName	The name of the resource manager.
RMID	The resource manager identifier. This is also logged in subsequent *Transaction Prepared* log records that record global transactions in which the resource manager is participating.
SwitchFile	The switch load file for this resource manager.
XAOpenString	The XA open string for this resource manager.
XACloseString	The XA close string for this resource manager.

Transaction Prepared

This log record is written by the XA Transaction Manager component of the queue manager. It indicates that the specified global transaction has been successfully prepared. Each of the participating resource managers will be instructed to commit. The *RMID* of each prepared resource manager is recorded in the log record. If the queue manager itself is participating in the transaction a *Participant Entry* with an *RMID* of zero will be present.

Transaction Forget

This log record is written by the XA Transaction Manager component of the queue manager. It follows the *Transaction Prepared* log record when the commit decision has been delivered to each participant.

Purge Queue

This logs the fact that all messages on a queue have been purged, for example, using the MQSC command CLEAR QUEUE.

Queue Attributes

This logs the initialization or change of the attributes of a queue.

Create Object

This logs the creation of a WebSphere MQ object.

ObjName	The name of the object that was created.
UserId	The user ID performing the creation.

Delete Object

This logs the deletion of a WebSphere MQ object.

ObjName	The name of the object that was deleted.

Parent topic: »Making sure that messages are not lost (logging)«

This build: January 26, 2011 11:55:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14810_

6.6.5. Backing up and restoring WebSphere® MQ

»Backing up queue managers and queue manager data.«

Periodically, you can take measures to protect queue managers against possible corruption caused by hardware failures. There are two ways of protecting a queue manager:

Backup the queue manager data

In the event of hardware failure, a queue manager can be forced to stop. If any queue manager log data is lost due to the hardware failure, the queue manager might be unable to restart. Through backing up queue manager data you may be able to recover some, or all, of the lost queue manager data.

In general, the more frequently you backup queue manager data, the less data you will lose in the event of hardware failure resulting in loss of integrity in the recovery log.

To backup queue manager data, the queue manager must not be running.

For instructions of how to backup queue manager data, and how to restore queue manager data, see:

- Backing up queue manager data.
- Restoring queue manager data.

Using a backup queue manager

In the event of severe hardware failure, a queue manager can be unrecoverable. In this situation, if the unrecoverable queue manager has a dedicated backup queue manager, the backup queue manager can be activated in place of the unrecoverable queue manager. If it was updated regularly, the backup queue manager log can contain log data up to, and including, the last complete log extent from the unrecoverable queue manager.

A backup queue manager can be updated while the existing queue manager is still running.

For instructions of how to create a backup queue manager, and how to activate a backup queue manager, see:

- Creating a backup queue manager.
- Starting a backup queue manager.

»Backing up queue manager data«
Backing up queue manager data can help you to guard against possible loss of data caused by hardware errors.

»Restoring queue manager data«

»Using a backup queue manager«

»Creating a backup queue manager«

»Updating a backup queue manager«

»Starting a backup queue manager«

Parent topic: »Availability, recovery and restart«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14690_

6.6.5.1. Backing up queue manager data

Backing up queue manager data can help you to guard against possible loss of data caused by hardware errors.

To take a backup copy of a queue manager's data:

1. Ensure that the queue manager is not running. If you try to take a backup of a running queue manager, the backup might not be consistent because of updates in progress when the files were copied.
   If possible, stop your queue manager in an orderly way. Try executing endmqm -w (a wait shutdown); only if that fails, use endmqm -i (an immediate shutdown).
2. Find the directories under which the queue manager places its data and its log files,

using the information in the configuration files. For more information about this, see Configuring WebSphere MQ.
Note: You might have some difficulty in understanding the names that appear in the directory. The names are transformed to ensure that they are compatible with the platform on which you are using WebSphere MQ. For more information about name transformations, see Understanding WebSphere MQ file names.

3. Take copies of all the queue manager's data and log file directories, including all subdirectories.
Make sure that you do not miss any files, especially the log control file, as described in What logs look like,and the configuration files (or equivalent Registry entries on Windows) as described in Initialization and configuration files. Some of the directories might be empty, but you need them all to restore the backup at a later date, so save them too.
4. Preserve the ownerships of the files. For WebSphere MQ for UNIX systems, you can do this with the tar command. (If you have queues larger than 2 GB, you cannot use tar; for more information, see Enabling large queues.)

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14700_

6.6.5.2. Restoring queue manager data

To restore a backup of a queue manager's data:

1. Ensure that the queue manager is not running.
2. Find the directories under which the queue manager places its data and its log files, using the information in the configuration files
3. Clear out the directories into which you are going to place the backed-up data.
4. Copy the backed-up queue manager data and log files into the correct places.
5. Update the configuration information files (or equivalent Registry entries on Windows).

Check the resulting directory structure to ensure that you have all the required directories.

See Directory structure on Windows systems and Directory content on UNIX and Linuxsystems for more information about WebSphere® MQ directories and subdirectories.

Make sure that you have a log control file as well as the log files. Also check that the WebSphere MQ and queue manager configuration files are consistent so that WebSphere MQ can look in the correct places for the restored data.

If the data was backed up and restored correctly, the queue manager will now start.

For circular logging, backup the queue manager data and log file directories at the same time as this should allow a consistent set of queue manager data and logs to be restored.

For linear logging, we recommend that you backup the queue manager data and log file directories at the same time. However, it is possible to restore only the queue manager data files if a corresponding complete sequence of log files is available.

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14710_

6.6.5.3. Using a backup queue manager

An existing queue manager can have a dedicated backup queue manager. A backup queue manager is an inactive copy of the existing queue manager. If the existing queue manager becomes unrecoverable due to severe hardware failure, the backup queue manager can be brought online to replace the unrecoverable queue manager.

The existing queue manager log files must regularly be copied to the backup queue manager to

ensure that the backup queue manager remains an effective method for disaster recovery. The existing queue manager does not need to be stopped for log files to be copied, however you should only copy a log file if the queue manager has finished writing to it. Because the existing queue manager log is continually updated, there is always a slight discrepancy between the existing queue manager log and the log data copied to the backup queue manager log. Regular updates to the backup queue manager minimizes the discrepancy between the two logs.

If a backup queue manager is required to be brought online it must be activated, and then started. The requirement to activate a backup queue manager before it is started is a preventative measure to protect against a backup queue manager being started accidentally. Once a backup queue manager is activated it can no longer be updated.

For information on how to create, update, and start a backup queue manager, see the following:

- Creating a backup queue manager
- Updating a backup queue manager
- Starting a backup queue manager

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14720_

6.6.5.4. Creating a backup queue manager

You can only use a backup queue manager when using linear logging.

To create a backup queue manager for an existing queue manager, do the following:

1. Create a backup queue manager for the existing queue manager using the control command crtmqm. The backup queue manager requires the following:
   - To have the same attributes as the existing queue manager, for example the queue manager name, the logging type, and the log file size.
   - To be on the same platform as the existing queue manager.
   - To be at an equal, or higher, code level than the existing queue manager.
2. Take copies of all the existing queue manager's data and log file directories, including all subdirectories, as described in Backing up queue manager data.
3. Overwrite the backup queue manager's data and log file directories, including all subdirectories, with the copies taken from the existing queue manager.
4. Execute the following control command on the backup queue manager:

   strmqm -r *BackupQMName*

   This flags the queue manager as a backup queue manager within WebSphere® MQ, and replays all the copied log extents to bring the backup queue manager in step with the existing queue manager.

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14730_

6.6.5.5. Updating a backup queue manager

»

To ensure that a backup queue manager remains an effective method for disaster recovery it must be updated regularly. Regular updating lessens the discrepancy between the backup queue manager log, and the current queue manager log. There is no need to stop the queue manager to be backed up.

To update a backup queue manager, do the following:

1. Issue the following Script (MQSC) command on the queue manager to be backed up:

```
RESET QMGR TYPE(ADVANCELOG)
```

This stops any writing to the current log, and then advances the queue manager logging to the next log extent. This ensures you backup all information logged up to the current time.

2. Obtain the (new) current active log extent number by issuing the following Script (MQSC) command on the queue manager to be backed up:

   ```
   DIS QMSTATUS CURRLOG
   ```

3. Copy the updated log extent files from the current queue manager log directory to the backup queue manager log directory - copy all the log extents since the last update, and up to (but not including) the current extent noted in step 2. Copy only log extent files, the ones beginning with "S...".

4. Issue the following control command on the backup queue manager:

   ```
   strmqm -r BackupQMName
   ```

   This replays all the copied log extents and brings the backup queue manager into step with the queue manager. When the replay finishes you receive a message that identifies all the log extents required for restart recovery, and all the log extents required for media recovery.
   Warning: If you copy a non-contiguous set of logs to the backup queue manager log directory, only the logs up to the point where the first missing log is found will be replayed.

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14740_

6.6.5.6. Starting a backup queue manager

To substitute an unrecoverable queue manager with its backup queue manager, do the following:

1. Execute the following control command to activate the backup queue manager:

   ```
   strmqm -a BackupQMName
   ```

   The backup queue manager is activated. Now active, the backup queue manager can no longer be updated.

2. Execute the following control command to start the backup queue manager:

   ```
   strmqm BackupQMName
   ```

   WebSphere MQ regards this as restart recovery, and utilizes the log from the backup queue manager. During the last update to the backup queue manager replay will have occurred, therefore only the active transactions from the last recorded checkpoint are rolled back.
   When an unrecoverable queue manager is substituted for a backup queue manager some of the queue manager data from the unrecoverable queue manager can be lost. The amount of lost data is dependent on how recently the backup queue manager was last updated. The more recently the last update, the less queue manager data loss.

3. Restart all channels.

Check the resulting directory structure to ensure that you have all the required directories.

See File system support for more information about WebSphere® MQ directories and subdirectories.

Make sure that you have a log control file as well as the log files. Also check that the WebSphere MQ and queue manager configuration files are consistent so that WebSphere MQ can look in the correct places for the restored data.

If the data was backed up and restored correctly, the queue manager will now start.

Note: Even though the queue manager data and log files are held in different directories, back up and restore the directories at the same time. If the queue manager data and log files have different ages, the queue manager is not in a valid state and will probably not start. If it does start, your data is likely to be corrupt.

Parent topic: »Backing up and restoring WebSphere MQ«

This build: January 26, 2011 11:55:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14750_

6.6.6. Recovery scenarios

Types of failures and how to recover from them.

This section looks at a number of possible problems and indicates how to recover from them.

Disk drive failures

Damaged queue manager object

Damaged single object
If a single object is reported as damaged during normal operation, for linear logging you can re-create the object from its media image. However, for circular logging you cannot re-create a single object.

Automatic media recovery failure

Parent topic: Availability, recovery and restart

This build: January 26, 2011 11:55:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14760_

6.6.6.1. Disk drive failures

You might have problems with a disk drive containing either the queue manager data, the log, or both. Problems can include data loss or corruption. The three cases differ only in the part of the data that survives, if any.

In *all* cases first check the directory structure for any damage and, if necessary, repair such damage. If you lose queue manager data, the queue manager directory structure might have been damaged. If so, re-create the directory tree manually before you restart the queue manager.

If damage has occurred to the queue manager data files, but not to the queue manager log files, then the queue manager will normally be able to restart. If any damage has occurred to the queue manager log files, then it is likely that the queue manager will not be able to restart.

Having checked for structural damage, there are a number of things you can do, depending on the type of logging that you use.

- Where there is major damage to the directory structure or any damage to the log, remove all the old files back to the QMgrName level, including the configuration files, the log, and the queue manager directory, restore the last backup, and restart the queue manager.
- For linear logging with media recovery, ensure that the directory structure is intact and restart the queue manager. If the queue manager restarts, check, using MQSC commands such as DISPLAY QUEUE, whether any other objects have been damaged. Recover those you find, using the rcrmqobj command. For example:

  ```
  rcrmqobj -m QMgrName -t all *
  ```

  where QMgrName is the queue manager being recovered. -t all * indicates that all damaged objects of any type are to be recovered. If only one or two objects have been reported as damaged, you can specify those objects by name and type here.
- For linear logging with media recovery and with an undamaged log, you might be able to restore a backup of the queue manager data leaving the existing log files and log control file unchanged. Starting the queue manager applies the changes from the log to bring the queue manager back to its state when the failure occurred.
  This method relies on two things:
  1. You must restore the checkpoint file as part of the queue manager data. This file

contains the information determining how much of the data in the log must be applied to give a consistent queue manager.

2. You must have the oldest log file required to start the queue manager at the time of the backup, and all subsequent log files, available in the log file directory.

If this is not possible, restore a backup of both the queue manager data and the log, both of which were taken at the same time. This causes message integrity to be lost.

- For circular logging, if the queue manager log files are damaged, restore the queue manager from the latest backup that you have. Once you have restored the backup, restart the queue manager and check as above for damaged objects. However, because you do not have media recovery, you must find other ways of re-creating the damaged objects.
  If the queue manager log files are not damaged, the queue manager will normally be able to restart. Following the restart you must identify all damaged objects, then delete and redefine them.

Parent topic: »Recovery scenarios«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14770_

6.6.6.2. Damaged queue manager object

If the queue manager object has been reported as damaged during normal operation, the queue manager performs a preemptive shutdown. There are two ways of recovering in these circumstances, depending on the type of logging you use:

- For linear logging, manually delete the file containing the damaged object and restart the queue manager. (You can use the dspmqfls command to determine the real, file-system name of the damaged object.) Media recovery of the damaged object is automatic.
- For circular logging, restore the last backup of the queue manager data and log, and restart the queue manager.

Parent topic: »Recovery scenarios«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14780_

6.6.6.3. Damaged single object

If a single object is reported as damaged during normal operation, for linear logging you can re-create the object from its media image. However, for circular logging you cannot re-create a single object.

Parent topic: »Recovery scenarios«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14790_

6.6.6.4. Automatic media recovery failure

If a local queue required for queue manager startup with a linear log is damaged, and the automatic media recovery fails, restore the last backup of the queue manager data and log and restart the queue manager.

Parent topic: »Recovery scenarios«

This build: January 26, 2011 11:55:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14800_

6.7. Problem determination

Problem determination is set of procedures to follow to handle problems experienced with WebSphere MQ.

This chapter suggests reasons for some of the problems you might experience using WebSphere® MQ. You usually start with a symptom, or set of symptoms, and trace them back to their cause.

Problem determination is not problem solving. However, the process of problem determination often enables you to solve a problem. For example, if you find that the cause of the problem is an error in an application program, you can solve the problem by correcting the error.

- Preliminary checks
- Looking at problems in more detail
- Application design considerations
- Error logs
- Dead-letter queues
- Configuration files and problem determination
- Tracing
- First-failure support technology (FFST)
- Problem determination with WebSphere MQ clients
- Java diagnostics
  For Java components of WebSphere MQ, for example the WebSphere MQ Explorer and the Java implementation of WebSphere MQ Transport for SOAP, diagnostic information is output using the standard WebSphere MQ diagnostic facilities or by Java diagnostic classes.

Parent topic: Configuring and managing WebSphere MQ

This build: January 26, 2011 11:55:16

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14820_

6.7.1. Preliminary checks

Before you start problem determination in detail, it is worth considering the facts to see if there is an obvious cause of the problem, or a likely area in which to start your investigation. This approach to debugging can often save a lot of work by highlighting a simple error, or by narrowing down the range of possibilities.

The cause of your problem could be in:

- WebSphere® MQ
- The network
- The application

The sections that follow raise some fundamental questions that you need to consider. As you work through the questions, make a note of anything that might be relevant to the problem. Even if your observations do not suggest a cause immediately, they could be useful later if you have to carry out a systematic problem determination exercise.

- Has WebSphere MQ run successfully before?
- Are there any error messages?
- Does the Windows Application Event Log show any WebSphere MQ errors? (Windows only)

»Are there any return codes explaining the problem?«

»Can you reproduce the problem?«

»Have any changes been made since the last successful run?«

»Has the application run successfully before?«

»Problems with commands«

»Does the problem affect specific parts of the network?«

»Does the problem occur at specific times of the day?«

»Is the problem intermittent?«

»Have you applied any maintenance updates?«

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:16

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14830_

6.7.1.1. Has WebSphere MQ run successfully before?

If WebSphere® MQ has not run successfully before, it is likely that you have not yet set it up correctly. See one of the following publications to check that you have installed the product correctly, and run the verification procedure:

- WebSphere MQ for AIX Quick Beginnings
- WebSphere MQ for HP-UX Quick Beginnings
- WebSphere MQ for Linux Quick Beginnings
- WebSphere MQ for Solaris Quick Beginnings
- WebSphere MQ for Windows Quick Beginnings

Also look at WebSphere MQ Intercommunication for information about post-installation configuration of WebSphere MQ.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:16

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14840_

6.7.1.2. Are there any error messages?

WebSphere® MQ uses error logs to capture messages concerning its own operation, any queue managers that you start, and error data coming from the channels that are in use. Check the error logs to see if any messages have been recorded that are associated with your problem.

See Error logs for information about the locations and contents of the error logs.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:16

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14850_

6.7.1.3. Does the Windows Application Event Log show any WebSphere MQ errors? (Windows only)

WebSphere® MQ also logs errors in the Windows Application Event Log. Some information about the cause of the error might be found here. To find the log, from the Computer Management panel, expand Event Viewer and select Application.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14860_

6.7.1.4. Are there any return codes explaining the problem?

If your application gets a return code indicating that a Message Queue Interface (MQI) call has failed, refer to the WebSphere MQ Application Programming Reference manual for a description of that return code.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14870_

6.7.1.5. Can you reproduce the problem?

If you can reproduce the problem, consider the conditions under which it is reproduced:

- Is it caused by a command or an equivalent administration request?
  Does the operation work if it is entered by another method? If the command works if it is entered on the command line, but not otherwise, check that the command server has not stopped, and that the queue definition of the SYSTEM.ADMIN.COMMAND.QUEUE has not been changed.
- Is it caused by a program? Does it fail on all WebSphere® MQ systems and all queue managers, or only on some?
- Can you identify any application that always seems to be running in the system when the problem occurs? If so, examine the application to see if it is in error.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14880_

6.7.1.6. Have any changes been made since the last successful run?

When you are considering changes that might recently have been made, think about the WebSphere® MQ system, and also about the other programs it interfaces with, the hardware, and any new applications. Consider also the possibility that a new application that you are not aware of might have been run on the system.

- Have you changed, added, or deleted any queue definitions?
- Have you changed or added any channel definitions? Changes might have been made to either WebSphere MQ channel definitions or any underlying communications definitions required by your application.
- Do your applications deal with return codes that they might get as a result of any changes you have made?
- Have you changed any component of the operating system that could affect the operation of WebSphere MQ? For example, have you modified the Windows Registry.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14890_

6.7.1.7. Has the application run successfully before?

If the problem appears to involve one particular application, consider whether the application has run successfully before.

Before you answer Yes to this question, consider the following:

- Have any changes been made to the application since it last ran successfully?
  If so, it is likely that the error lies somewhere in the new or modified part of the application. Take a look at the changes and see if you can find an obvious reason for the problem. Is it possible to retry using a back level of the application?
- Have all the functions of the application been fully exercised before?
  Could it be that the problem occurred when part of the application that had never been invoked before was used for the first time? If so, it is likely that the error lies in that part of the application. Try to find out what the application was doing when it failed, and check the source code in that part of the program for errors.
  If a program has been run successfully on many previous occasions, check the current queue status and the files that were being processed when the error occurred. It is possible that they contain some unusual data value that invokes a rarely-used path in the program.
- Does the application check all return codes?
  Has your WebSphere® MQ system been changed, perhaps in a minor way, such that your application does not check the return codes it receives as a result of the change. For example, does your application assume that the queues it accesses can be shared? If a queue has been redefined as exclusive, can your application deal with return codes indicating that it can no longer access that queue?
- Does the application run on other WebSphere MQ systems?
  Could it be that there is something different about the way that this WebSphere MQ system is set up that is causing the problem? For example, have the queues been defined with the same message length or priority?

»If the application has not run successfully before«

»Common programming errors«

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14900_

6.7.1.7.1. If the application has not run successfully before

If your application has not yet run successfully, examine it carefully to see if you can find any errors.

Before you look at the code, and depending upon which programming language the code is written in, examine the output from the translator, or the compiler and linkage editor, to see if any errors have been reported.

If your application fails to translate, compile, or link-edit into the load library, it will also fail to run if you attempt to invoke it. See the WebSphere MQ Application Programming Guide for information about building your application.

If the documentation shows that each of these steps was accomplished without error, consider the coding logic of the application. Do the symptoms of the problem indicate the function that is failing and, therefore, the piece of code in error? See Common programming errors for some examples of common errors that cause problems with WebSphere® MQ applications.

Parent topic: »Has the application run successfully before?«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14910_

6.7.1.7.2. Common programming errors

The errors in the following list illustrate the most common causes of problems encountered while running WebSphere® MQ programs. Consider the possibility that the problem with your WebSphere MQ system could be caused by one or more of these errors:

- Assuming that queues can be shared, when they are in fact exclusive.
- Passing incorrect parameters in an MQI call.
- Passing insufficient parameters in an MQI call. This might mean that WebSphere MQ cannot set up completion and reason codes for your application to process.
- Failing to check return codes from MQI requests.
- Passing variables with incorrect lengths specified.
- Passing parameters in the wrong order.
- Failing to initialize *MsgId* and *CorrelId* correctly.
- Failing to initialize *Encoding* and *CodedCharSetId* following MQRC_TRUNCATED_MSG_ACCEPTED.

Parent topic: »Has the application run successfully before?«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14920_

6.7.1.8. Problems with commands

Be careful when including special characters, for example, back slash (\) and double quote (") characters, in descriptive text for some commands. If you use either of these characters in descriptive text, precede them with a \, that is, enter \\ or \" if you want \ or " in your text.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14930_

6.7.1.9. Does the problem affect specific parts of the network?

You might be able to identify specific parts of the network that are affected by the problem (remote queues, for example). If the link to a remote message queue manager is not working, the messages cannot flow to a remote queue.

Check that the connection between the two systems is available, and that the intercommunication component of WebSphere® MQ has started.

Check that messages are reaching the transmission queue, and check the local queue definition of the transmission queue and any remote queues.

Have you made any network-related changes, or changed any WebSphere MQ definitions, that might account for the problem?

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14940_

6.7.1.10. Does the problem occur at specific times of the day?

If the problem occurs at specific times of day, it could be that it depends on system loading. Typically, peak system loading is at mid-morning and mid-afternoon, so these are the times when load-dependent problems are most likely to occur. (If your WebSphere® MQ network extends across more than one time zone, peak system loading might seem to occur at some other time of day.)

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14950_

6.7.1.11. Is the problem intermittent?

An intermittent problem could be caused by the way that processes can run independently of each other. For example, a program might issue an MQGET call without specifying a wait option before an earlier process has completed. An intermittent problem might also be seen if your application tries to get a message from a queue while the call that put the message is in-doubt (that is, before it has been committed or backed out).

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14960_

6.7.1.12. Have you applied any maintenance updates?

If you have applied a maintenance update to WebSphere® MQ, check that the update action completed successfully and that no error message was produced.

- Did the update have any special instructions?
- Was any test run to verify that the update was applied correctly and completely?
- Does the problem still exist if WebSphere MQ is restored to the previous maintenance level?
- If the installation was successful, check with the IBM® Support Center for any maintenance package errors.
- If a maintenance package has been applied to any other program, consider the effect it might have on the way WebSphere MQ interfaces with it.

Parent topic: »Preliminary checks«

This build: January 26, 2011 11:55:17

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14970_

6.7.2. Looking at problems in more detail

Perhaps the preliminary checks have enabled you to find the cause of the problem. If so, you should now be able to resolve it, possibly with the help of other books in the WebSphere® MQ library and in the libraries of other licensed programs.

If you have not yet found the cause, start to look at the problem in greater detail. The purpose of this section is to help you identify the cause of your problem if the preliminary checks have not enabled you to find it. When you have established that no changes have been made to your system, and that there are no problems with your application programs, choose the option that best describes the symptoms of your problem.

- Have you obtained incorrect output?

- Have you failed to receive a response from a PCF command?
- Are some of your queues failing?
- Are you receiving an error code when creating or starting a queue manager? (Windows only)
- Does the problem affect only remote queues?
- Is your application or system running slowly?

If none of these symptoms describe your problem, consider whether it might have been caused by another component of your system.

Have you obtained incorrect output?

Have you failed to receive a response from a PCF command?

Are some of your queues failing?

Are you receiving an error code when creating or starting a queue manager? (Windows only)

Does the problem affect only remote queues?

Is your application or system running slowly?

Parent topic: Problem determination

This build: January 26, 2011 11:55:17

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14980_

6.7.2.1. Have you obtained incorrect output?

In this book, *incorrect output* refers to your application:

- Not receiving a message that it was expecting.
- Receiving a message containing unexpected or corrupted information.
- Receiving a message that it was not expecting, for example, one that was destined for a different application.

Messages that do not appear on the queue

Messages that contain unexpected or corrupted information

Problems with incorrect output when using distributed queues

Parent topic: Looking at problems in more detail

This build: January 26, 2011 11:55:17

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa14990_

6.7.2.1.1. Messages that do not appear on the queue

If messages do not appear when you are expecting them, check for the following:

- Has the message been put on the queue successfully?
  - Has the queue been defined correctly? For example, is MAXMSGL sufficiently large?
  - Is the queue enabled for putting?
  - Is the queue already full?
  - Has another application got exclusive access to the queue?
- Are you able to get any messages from the queue?

- Do you need to take a sync point?
  If messages are being put or retrieved within sync point, they are not available to other tasks until the unit of recovery has been committed.
- Is your wait interval long enough?
  You can set the wait interval as an option for the MQGET call. Ensure that you are waiting long enough for a response.
- Are you waiting for a specific message that is identified by a message or correlation identifier (*MsgId* or *CorrelId*)?
  Check that you are waiting for a message with the correct *MsgId* or *CorrelId*. A successful MQGET call sets both these values to that of the message retrieved, so you might need to reset these values in order to get another message successfully.
  Also, check whether you can get other messages from the queue.
- Can other applications get messages from the queue?
- Was the message you are expecting defined as persistent?
  If not, and WebSphere® MQ has been restarted, the message has been lost.
- Has another application got exclusive access to the queue?

If you cannot find anything wrong with the queue, and WebSphere MQ is running, check the process that you expected to put the message onto the queue for the following:

- Did the application start?
  If it should have been triggered, check that the correct trigger options were specified.
- Did the application stop?
- Is a trigger monitor running?
- Was the trigger process defined correctly?
- Did the application complete correctly?
  Look for evidence of an abnormal end in the job log.
- Did the application commit its changes, or were they backed out?

If multiple transactions are serving the queue, they can conflict with one another. For example, suppose one transaction issues an MQGET call with a buffer length of zero to find out the length of the message, and then issues a specific MQGET call specifying the *MsgId* of that message. However, in the meantime, another transaction issues a successful MQGET call for that message, so the first application receives a reason code of MQRC_NO_MSG_AVAILABLE. Applications that are expected to run in a multiple server environment must be designed to cope with this situation.

Consider that the message could have been received, but that your application failed to process it in some way. For example, did an error in the expected format of the message cause your program to reject it? If this is the case, refer to Messages that contain unexpected or corrupted information.

Parent topic: »Have you obtained incorrect output?«

This build: January 26, 2011 11:55:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15000_

6.7.2.1.2. Messages that contain unexpected or corrupted information

If the information contained in the message is not what your application was expecting, or has been corrupted in some way, consider the following:

- Has your application, or the application that put the message onto the queue, changed?
  Ensure that all changes are simultaneously reflected on all systems that need to be aware of the change.
  For example, the format of the message data might have been changed, in which case, both applications must be recompiled to pick up the changes. If one application has not been recompiled, the data will appear corrupt to the other.
- Is an application sending messages to the wrong queue?
  Check that the messages your application is receiving are not intended for an application servicing a different queue. If necessary, change your security definitions to prevent unauthorized applications from putting messages on to the wrong queues.
  If your application uses an alias queue, check that the alias points to the correct queue.
- Has the trigger information been specified correctly for this queue?

Check that your application should have started; or should a different application have started?

If these checks do not enable you to solve the problem, check your application logic, both for the program sending the message, and for the program receiving it.

Parent topic: »Have you obtained incorrect output?«

This build: January 26, 2011 11:55:18

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15010_

6.7.2.1.3. Problems with incorrect output when using distributed queues

If your application uses distributed queues, consider the following points:

- Has WebSphere® MQ been correctly installed on both the sending and receiving systems, and correctly configured for distributed queuing?
- Are the links available between the two systems?
  Check that both systems are available, and connected to WebSphere MQ. Check that the connection between the two systems is active.
  You can use the MQSC command PING against either the queue manager (PING QMGR) or the channel (PING CHANNEL) to verify that the link is operable.
- Is triggering set on in the sending system?
- Is the message for which you are waiting a reply message from a remote system?
  Check that triggering is activated in the remote system.
- Is the queue already full?
  If so, check if the message has been put onto the dead-letter queue.
  The dead-letter queue header contains a reason or feedback code explaining why the message could not be put onto the target queue. See the WebSphere MQ Application Programming Reference for information about the dead-letter queue header structure.
- Is there a mismatch between the sending and receiving queue managers?
  For example, the message length could be longer than the receiving queue manager can handle.
- Are the channel definitions of the sending and receiving channels compatible?
  For example, a mismatch in sequence number wrap can stop the distributed queuing component. See WebSphere MQ Intercommunication for more information about distributed queuing.
- Is data conversion involved? If the data formats between the sending and receiving applications differ, data conversion is necessary. Automatic conversion occurs when the MQGET call is issued if the format is recognized as one of the built-in formats.
  If the data format is not recognized for conversion, the data conversion exit is taken to allow you to perform the translation with your own routines.
  Refer to the WebSphere MQ Application Programming Guide for further details of data conversion.

Parent topic: »Have you obtained incorrect output?«

This build: January 26, 2011 11:55:18

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15020_

6.7.2.2. Have you failed to receive a response from a PCF command?

If you have issued a command but have not received a response, consider the following:

- Is the command server running?
  Work with the dspmqcsv command to check the status of the command server.
  - If the response to this command indicates that the command server is not running, use the strmqcsv command to start it.
  - If the response to the command indicates that the SYSTEM.ADMIN.COMMAND.QUEUE is not enabled for MQGET requests, enable the queue for MQGET requests.

- Has a reply been sent to the dead-letter queue?
  The dead-letter queue header structure contains a reason or feedback code describing the problem. See the WebSphere MQ Application Programming Reference for information about the dead-letter queue header structure (MQDLH).
  If the dead-letter queue contains messages, you can use the provided browse sample application (amqsbcg) to browse the messages using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for all the messages on the named queue.
- Has a message been sent to the error log?
  See Error logs for further information.
- Are the queues enabled for put and get operations?
- Is the *WaitInterval* long enough?
  If your MQGET call has timed out, a completion code of MQCC_FAILED and a reason code of MQRC_NO_MSG_AVAILABLE are returned. (See the WebSphere MQ Application Programming Reference for information about the *WaitInterval* field, and completion and reason codes from MQGET.)
- If you are using your own application program to put commands onto the SYSTEM.ADMIN.COMMAND.QUEUE, do you need to take a sync point?
  Unless you have specifically excluded your request message from sync point, you need to take a sync point before receiving reply messages.
- Are the MAXDEPTH and MAXMSGL attributes of your queues set sufficiently high?
- Are you using the *CorrelId* and *MsgId* fields correctly?
  Set the values of *MsgId* and *CorrelId* in your application to ensure that you receive all messages from the queue.

Try stopping the command server and then restarting it, responding to any error messages that are produced.

If the system still does not respond, the problem could be with either a queue manager or the whole of the WebSphere® MQ system. First, try stopping individual queue managers to isolate a failing queue manager. If this does not reveal the problem, try stopping and restarting WebSphere MQ, responding to any messages that are produced in the error log.

If the problem still occurs after restart, contact your IBM® Support Center for help.

Parent topic: »Looking at problems in more detail«

This build: January 26, 2011 11:55:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15030_

6.7.2.3. Are some of your queues failing?

If you suspect that the problem occurs with only a subset of queues, check the local queues that you think are having problems:

1. Display the information about each queue. You can use the MQSC command DISPLAY QUEUE to display the information.
2. Use the data displayed to do the following checks:
   - If CURDEPTH is at MAXDEPTH, the queue is not being processed. Check that all applications are running normally.
   - If CURDEPTH is not at MAXDEPTH, check the following queue attributes to ensure that they are correct:
     - If triggering is being used:
       - Is the trigger monitor running?
       - Is the trigger depth too great? That is, does it generate a trigger event often enough?
       - Is the process name correct?
       - Is the process available and operational?
     - Can the queue be shared? If not, another application could already have it open for input.
     - Is the queue enabled appropriately for GET and PUT?

- If there are no application processes getting messages from the queue, determine why this is so. It could be because the applications need to be started, a connection has been disrupted, or the MQOPEN call has failed for some reason.
  Check the queue attributes IPPROCS and OPPROCS. These attributes indicate whether the queue has been opened for input and output. If a value is zero, it indicates that no operations of that type can occur. The values might have changed; the queue might have been open but is now closed.
  You need to check the status at the time you expect to put or get a message.

If you are unable to solve the problem, contact your IBM® Support Center for help.

Parent topic: »Looking at problems in more detail«

This build: January 26, 2011 11:55:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15040_

6.7.2.4. Are you receiving an error code when creating or starting a queue manager? (Windows only)

If the WebSphere® MQ Explorer, or the amqmdain command, fails to create or start a queue manager, indicating an authority problem, it might be because the user under which the "IBM® MQSeries® Services" process (AMQMSRVN) is running has insufficient rights.

Ensure that the user with which the AMQMSRVN service is configured has the rights described in User rights required for AMQMSRVN. By default this service is configured to run as the MUSR_MQADMIN user.

Parent topic: »Looking at problems in more detail«

This build: January 26, 2011 11:55:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15050_

6.7.2.5. Does the problem affect only remote queues?

If the problem affects only remote queues:

- Check that required channels have started, can be triggered, and any required initiators are running.
- Check that the programs that should be putting messages to the remote queues have not reported problems.
- If you use triggering to start the distributed queuing process, check that the transmission queue has triggering set on. Also, check that the trigger monitor is running.
- Check the error logs for messages indicating channel errors or problems.
- If necessary, start the channel manually. See WebSphere MQ Intercommunication for information about starting channels.

Parent topic: »Looking at problems in more detail«

This build: January 26, 2011 11:55:18

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15060_

6.7.2.6. Is your application or system running slowly?

If your application is running slowly, it might be in a loop or waiting for a resource that is not available.

This might also indicate a performance problem. Perhaps your system is operating near the limits of its capacity. This type of problem is probably worst at peak system load times,

typically at mid-morning and mid-afternoon. (If your network extends across more than one time zone, peak system load might seem to occur at some other time.)

A performance problem might be caused by a limitation of your hardware.

If you find that performance degradation is not dependent on system loading, but happens sometimes when the system is lightly loaded, a poorly-designed application program is probably to blame. This could appear to be a problem that only occurs when certain queues are accessed.

The following symptoms might indicate that WebSphere® MQ is running slowly:

- Your system is slow to respond to MQSC commands.
- Repeated displays of the queue depth indicate that the queue is being processed slowly for an application with which you would expect a large amount of queue activity.

If the performance of your system is still degraded after reviewing the above possible causes, the problem might lie with WebSphere MQ itself. If you suspect this, contact your IBM® Support Center for help.

»Tuning performance for nonpersistent messages on AIX«
If you are using AIX® V5.3 or later, consider setting your tuning parameter to exploit full performance for nonpersistent messages.

Parent topic: »Looking at problems in more detail«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15070_

6.7.2.6.1. Tuning performance for nonpersistent messages on AIX

If you are using AIX® V5.3 or later, consider setting your tuning parameter to exploit full performance for nonpersistent messages.

To set the tuning parameter so that it takes effect immediately, issue the following command as a root user:

```
/usr/sbin/ioo -o j2_nPagesPerWriteBehindCluster=0
```

To set the tuning parameter so that it takes effect immediately and persists over reboots, issue the following command as a root user:

```
/usr/sbin/ioo -p -o j2_nPagesPerWriteBehindCluster=0
```

Normally, nonpersistent messages are kept only in memory, but there are circumstances where AIX can schedule nonpersistent messages to be written to disk. Messages scheduled to be written to disk are unavailable for MQGET until the disk write completes. The suggested tuning command varies this threshold; instead of scheduling messages to be written to disk when 16 kilobytes of data are queued, the write-to-disk occurs only when real storage on the machine becomes close to full. This is a global alteration and may effect other software components.

On AIX, when using multithreaded applications and especially when running on machines with multiple CPUs, we strongly recommend setting AIXTHREAD_SCOPE=S in the environment before starting the application, for better performance and more solid scheduling. For example:

```
export AIXTHREAD_SCOPE=S
```

Setting AIXTHREAD_SCOPE=S means that user threads created with default attributes will be placed into system-wide contention scope. If a user thread is created with system-wide contention scope, it is bound to a kernel thread and it is scheduled by the kernel. The underlying kernel thread is not shared with any other user thread.

Parent topic: »Is your application or system running slowly?«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15080_

6.7.3. Application design considerations

There are a number of ways in which poor program design can affect performance. These can be difficult to detect because the program can appear to perform well itself, but affect the performance of other tasks. Several problems specific to programs making WebSphere® MQ calls are discussed in the following sections.

For more information about application design, see the WebSphere MQ Application Programming Guide.

- Effect of message length
- Effect of message persistence
- Searching for a particular message
- Queues that contain messages of different lengths
- Frequency of sync points
- Use of the MQPUT1 call
- Number of threads in use

Parent topic: Problem determination

This build: January 26, 2011 11:55:19

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15090_

6.7.3.1. Effect of message length

The amount of data in a message can affect the performance of the application that processes the message. To achieve the best performance from your application, send only the essential data in a message. For example, in a request to debit a bank account, the only information that might need to be passed from the client to the server application is the account number and the amount of the debit.

Parent topic: Application design considerations

This build: January 26, 2011 11:55:19

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15100_

6.7.3.2. Effect of message persistence

Persistent messages are usually logged. Logging messages reduces the performance of your application, so use persistent messages for essential data only. If the data in a message can be discarded if the queue manager stops or fails, use a nonpersistent message.

Parent topic: Application design considerations

This build: January 26, 2011 11:55:19

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15110_

6.7.3.3. Searching for a particular message

The MQGET call usually retrieves the first message from a queue. If you use the message and correlation identifiers (*MsgId* and *CorrelId*) in the message descriptor to specify a particular message, the queue manager has to search the queue until it finds that message. Using the MQGET call in this way affects the performance of your application.

Parent topic: »Application design considerations«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15120_

6.7.3.4. Queues that contain messages of different lengths

If your application cannot use messages of a fixed length, grow and shrink the buffers dynamically to suit the typical message size. If the application issues an MQGET call that fails because the buffer is too small, the size of the message data is returned. Add code to your application so that the buffer is resized accordingly and the MQGET call is re-issued.

Note: if you do not set the *MaxMsgLength* attribute explicitly, it defaults to 4 MB, which might be very inefficient if this is used to influence the application buffer size.

Parent topic: »Application design considerations«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15130_

6.7.3.5. Frequency of sync points

Programs that issue very large numbers of MQPUT or MQGET calls within sync point, without committing them, can cause performance problems. Affected queues can fill up with messages that are currently inaccessible, while other tasks might be waiting to get these messages. This has implications in terms of storage, and in terms of threads tied up with tasks that are attempting to get messages.

Parent topic: »Application design considerations«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15140_

6.7.3.6. Use of the MQPUT1 call

Use the MQPUT1 call only if you have a single message to put on a queue. If you want to put more than one message, use the MQOPEN call, followed by a series of MQPUT calls and a single MQCLOSE call.

Parent topic: »Application design considerations«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15150_

6.7.3.7. Number of threads in use

For WebSphere® MQ for Windows, an application might require a large number of threads. Each queue manager process is allocated a maximum allowable number of application threads.

Applications might use too many threads. Consider whether the application takes into account this possibility and that it takes actions either to stop or to report this type of occurrence.

Parent topic: »Application design considerations«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15160_

6.7.4. Error logs

WebSphere® MQ uses a number of error logs to capture messages concerning its own operation of WebSphere MQ, any queue managers that you start, and error data coming from the channels that are in use.

The location of the error logs depends on whether the queue manager name is known and whether the error is associated with a client.

- If the queue manager name is known, the location of the error log is shown in Table 1.

  Table 1. Queue manager error log directory

Platform	Directory
UNIX systems	/var/mqm/qmgrs/*qmname*/errors
Windows systems	c:\Program Files\IBM\WebSphere MQ\qmgrs*qmname*\errors

- If the queue manager name is not known, the location of the error log is shown in Table 2.

  Table 2. System error log directory

Platform	Directory
UNIX systems	/var/mqm/errors
Windows systems	c:\Program Files\IBM\WebSphere MQ\errors

- If an error has occurred with a client application, the location of the error log on the client is shown in Table 3.

  Table 3. Client error log directory

Platform	Directory
UNIX systems	/var/mqm/errors
Windows systems	c:\Program Files\IBM\WebSphere MQ Client\errors

In WebSphere MQ for Windows, an indication of the error is also added to the Application Log, which can be examined with the Event Viewer application provided with Windows systems.

»Error log files«

»Error log access restrictions under UNIX systems«

»Ignoring error codes under UNIX systems«

»Ignoring error codes under Windows systems«

»Operator messages«

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15170_

6.7.4.1. Error log files

At installation time an errors subdirectory is created in the /var/mqm file path under UNIX systems, and in the \IBM\WebSphere MQ\ file path under Windows systems. The errors subdirectory can contain up to three error log files named:

- AMQERR01.LOG
- AMQERR02.LOG
- AMQERR03.LOG

After you have created a queue manager, it creates three error log files when it needs them. These files have the same names as those in the system error log directory, that is AMQERR01, AMQERR02, and AMQERR03, and each has a default capacity of 256 KB. The capacity can be altered in the Extended queue manager properties page from the WebSphere® MQ Explorer, or in the QMErrorLog stanza in the qm.ini file. These files are placed in the errors subdirectory in the /var/mqm/qmgrs/*qmname* file path under UNIX systems, or in the \IBM\WebSphere MQ\qmgrs*qmname*\errors file path under Windows systems.

As error messages are generated, they are placed in AMQERR01. When AMQERR01 gets bigger than 256 KB it is copied to AMQERR02. Before the copy, AMQERR02 is copied to AMQERR03.LOG. The previous contents, if any, of AMQERR03 are discarded.

The latest error messages are thus always placed in AMQERR01, the other files being used to maintain a history of error messages.

All messages relating to channels are also placed in the appropriate queue manager's errors files, unless the queue manager is unavailable, or its name is unknown, in which case channel-related messages are placed in the system error log directory.

To examine the contents of any error log file, use your usual system editor.

»Early errors«

»An example of an error log«

Parent topic: »Error logs«

This build: January 26, 2011 11:55:19

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15180_

6.7.4.1.1. Early errors

There are a number of special cases where these error logs have not yet been established and an error occurs. WebSphere® MQ attempts to record any such errors in an error log. The location of the log depends on how much of a queue manager has been established.

If, because of a corrupt configuration file for example, no location information can be determined, errors are logged to an errors directory that is created at installation time on the root directory (/var/mqm or C:\Program Files\IBM\WebSphere MQ).

If WebSphere MQ can read its configuration information, and can access the value for the Default Prefix, errors are logged in the errors subdirectory of the directory identified by the Default Prefix attribute. For example, if the default prefix is C:\Program Files\IBM\WebSphere MQ, errors are logged in C:\Program Files\IBM\WebSphere MQ\errors.

For further information about configuration files, see Configuring WebSphere MQ.

Note: Errors in the Windows Registry are notified by messages when a queue manager is started.

Parent topic: »Error log files«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15190_

6.7.4.1.2. An example of an error log

Figure 1 shows an extract from a WebSphere® MQ error log:
Figure 1. Sample WebSphere MQ error log

```
17/11/2004  10:32:29 - Process(2132.1) User(USER_1) Program(runmqchi.exe)
AMQ9542: Queue manager is ending.

EXPLANATION:
The program will end because the queue manager is quiescing.
```

```
ACTION:
None.
----- amqrimna.c : 931 --------------------------------------------------------
```

Parent topic: »Error log files«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15200_

6.7.4.2. Error log access restrictions under UNIX systems

Certain error log directories and error logs have access restrictions under UNIX systems. To gain the following access permissions, a user or application must be a member of the mqm group:

- Read and write access to all queue manager error log directories.
- Read and write access to all queue manager error logs.
- Write access to the system error logs.

If an unauthorized user or application attempts to write a message to a queue manager error log directory, the message is redirected to the system error log directory.

Parent topic: »Error logs«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15210_

6.7.4.3. Ignoring error codes under UNIX systems

On UNIX systems, if you do not want certain error messages to be written to a queue manager error log, you can specify the error codes that are to be ignored using the QMErrorLog stanza. For more information, see Queue manager error logs.

Parent topic: »Error logs«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15220_

6.7.4.4. Ignoring error codes under Windows systems

On Windows systems, if an error message has a severity of ERROR, the message is written to both the WebSphere® MQ error log and the Windows Application Event Log.

If you do not want certain error messages to be written to the Windows Application Event Log, you can specify the error codes that are to be ignored in the Windows registry. Use the registry key:

```
HKEY_LOCAL_MACHINE\Software\IBM\MQSeries\CurrentVersion\IgnoredErrorCodes
```

The value that you set it to is an array of strings delimited by the NULL character, with each string value relating to the error code that you want ignored from the error log. The complete list is terminated with a NULL character, which is of type REG_MULTI_SZ.

For example, if you want WebSphere MQ to exclude error codes AMQ3045, AMQ6055, and AMQ8079 from the Windows Application Event Log, set the value to:

```
AMQ3045\0AMQ6055\0AMQ8079\0\0
```

The list of messages you want to exclude is defined for all queue managers on the machine. Any changes you make to the configuration will not take effect until each queue manager is restarted.

Parent topic: »Error logs«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15230_

6.7.4.5. Operator messages

Operator messages identify normal errors, typically caused directly by users doing things like using parameters that are not valid on a command. Operator messages are national-language enabled, with message catalogs installed in standard locations.

These messages are written to the associated window, if any. In addition, some operator messages are written to the AMQERR01.LOG file in the queue manager directory, and others to the equivalent file in the system error log directory.

Parent topic: »Error logs«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15240_

6.7.5. Dead-letter queues

Messages that cannot be delivered for some reason are placed on the dead-letter queue. You can check whether the queue contains any messages by issuing the MQSC command DISPLAY QUEUE. If the queue contains messages, use the provided browse sample application (amqsbcg) to browse messages on the queue using the MQGET call. The sample application steps through all the messages on a named queue for a named queue manager, displaying both the message descriptor and the message context fields for each message. See Browsing queues for more information about running this sample and about the kind of output it produces.

You must decide how to dispose of any messages found on the dead-letter queue, depending on the reasons for the messages being put on the queue.

Problems might occur if you do not associate a dead-letter queue with each queue manager. For more information about dead-letter queues, see The WebSphere MQ dead-letter queue handler.

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15250_

6.7.6. Configuration files and problem determination

Configuration file errors typically prevent queue managers from being found, and result in *queue manager unavailable* errors. Ensure that the configuration files exist, and that the WebSphere® MQ configuration file references the correct queue manager and log directories.

Errors in the Windows Registry are notified by messages when a queue manager is started.

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15260_

6.7.7. Tracing

This section describes how to produce a trace for WebSphere® MQ.

»Tracing WebSphere MQ for Windows«

»Tracing WebSphere MQ for UNIX systems«
This is a brief introduction to the trace control commands you can use in Websphere MQ for UNIX.

»Trace files«
Introduction to trace files for UNIX and Linux systems; their file paths and their file names.

»Tracing Secure Sockets Layer (SSL) iKeyman and iKeycmd functions«

»Tracing with the AIX system trace«

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15270_

6.7.7.1. Tracing WebSphere MQ for Windows

You enable or modify tracing using the strmqtrc control command, described in strmqtrc. To stop tracing, you use the endmqtrc control command, described in endmqtrc.

In WebSphere® MQ for Windows systems, you can also start and stop tracing using the WebSphere MQ Explorer, as follows:

1. Start the WebSphere MQ Explorer from the Start menu.
2. In the Navigator View, right-click the WebSphere MQ tree node, and select Trace.... The Trace Dialog is displayed.
3. Click Start or Stop as appropriate.

»Selective component tracing on WebSphere MQ for Windows«

Selective process tracing on WebSphere MQ for Windows
This introduces the concept of selective process tracing on Websphere MQ for Windows.

»Trace files«
This introduces trace files for Windows systems; their file paths and their file names.

»An example of WebSphere MQ for Windows trace data«

Parent topic: »Tracing«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15280_

6.7.7.1.1. Selective component tracing on WebSphere® MQ for Windows

Use the -t and -x options to control the amount of trace detail to record. By default, all trace points are enabled. The -x option enables you to specify the points that you do not want to trace. So if, for example, you want to trace only data flowing over communications networks, use:

```
strmqtrc -x all -t comms
```

For a full description of the trace command, see strmqtrc.
Parent topic: »Tracing WebSphere MQ for Windows«

This build: January 26, 2011 11:55:20

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15290_

6.7.7.1.2. Selective process tracing on WebSphere® MQ for Windows

This introduces the concept of selective process tracing on Websphere MQ for Windows.

Use the -p option of the strmqtrc command control to restrict trace generation to specified named processes. For example, to trace all threads that result from any executing process called amqxxx.exe, use the following command:

```
strmqtrc -p amqxxx.exe
```

For a full description of the trace command, see strmqtrc.
Parent topic: »Tracing WebSphere MQ for Windows«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15295_

6.7.7.1.3. Trace files

This introduces trace files for Windows systems; their file paths and their file names.

By default, the trace files are placed in the directory\<mqmwork>\trace, where <mqmwork> is the directory selected, when WebSphere® MQ is installed, to hold WebSphere MQ data files. If you want to, you can specify a different directory path when you start a trace.

Trace files are named AMQ*ppppp.qq*.TRC, where:

ppppp	Is the ID of the process reporting the error.
qq	Starts at 0. If the full file name already exists, this value is incremented by one until a unique trace file name is found. A trace file name can already exist if a process is reused.

Note:

1. The process identifier can contain fewer, or more, digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

Parent topic: »Tracing WebSphere MQ for Windows«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15300_

6.7.7.1.4. An example of WebSphere MQ for Windows trace data

»Figure 1 shows an extract from a WebSphere® MQ for Windows trace:
Figure 1. Sample WebSphere MQ for Windows trace

```
   Counter   TimeStamp            PID.TID     Ident          Data
   ============================================================

   00000EF7 16:18:56.381367    2512.1             :        !! - Thread stack
   00000EF8 16:18:56.381406    2512.1             :        !! - -> InitProcessInitialisation
   00000EF9 16:18:56.381429    2512.1             :        --{  InitProcessInitialisation
   00000EFA 16:18:56.381514    2512.1             :        ---{  xcsReleaseThreadMutexSem
   00000EFB 16:18:56.381529    2512.1             :        ---}  xcsReleaseThreadMutexSem (rc=O
   00000EFC 16:18:56.381540    2512.1             :        ---{  xcsGetEnvironmentString
   00000EFD 16:18:56.381574    2512.1             :        xcsGetEnvironmentString[AMQ_REUSE_SHA
   00000EFE 16:18:56.381587    2512.1             :        ---}! xcsGetEnvironmentString (rc=xe
   00000EFF 16:18:56.381612    2512.1             :        ---{  xcsGetEnvironmentInteger
   00000F00 16:18:56.381622    2512.1             :        ----{  xcsGetEnvironmentString
   00000F01 16:18:56.381647    2512.1             :        xcsGetEnvironmentString[AMQ_AFFINITY_
   00000F02 16:18:56.381660    2512.1             :        ----}! xcsGetEnvironmentString (rc=xe
```

```
00000F03 16:18:56.381673   2512.1            :        ---}! xcsGetEnvironmentInteger (rc=xe
00000F04 16:18:56.381684   2512.1            :        ---{  xcsGetEnvironmentString
00000F05 16:18:56.381708   2512.1            :        xcsGetEnvironmentString[AMQ_FFSTINFO]
00000F06 16:18:56.381747   2512.1            :        ---}! xcsGetEnvironmentString (rc=xe
00000F07 16:18:56.381760   2512.1            :        ---{  xcsIsEnvironment
00000F08 16:18:56.381783   2512.1            :        xcsIsEnvironment[AMQ_DEBUG_MTIME] = 
00000F09 16:18:56.381793   2512.1            :        ---}  xcsIsEnvironment (rc=OK)
00000F0A 16:18:56.381804   2512.1            :        ---{  xcsGetEnvironmentInteger
00000F0B 16:18:56.381811   2512.1            :        ----{  xcsGetEnvironmentString
00000F0C 16:18:56.381835   2512.1            :        xcsGetEnvironmentString[AMQ_CBM_REUSE
00000F0D 16:18:56.381848   2512.1            :        ----}! xcsGetEnvironmentString (rc=xe
00000F0E 16:18:56.381861   2512.1            :        ---}! xcsGetEnvironmentInteger (rc=xe
00000F0F 16:18:56.381874   2512.1            :        ---{  xcsGetEnvironmentInteger
00000F10 16:18:56.381885   2512.1            :        ----{  xcsGetEnvironmentString
00000F11 16:18:56.381908   2512.1            :        xcsGetEnvironmentString[AMQ_CBM_MAX_C
00000F12 16:18:56.381919   2512.1            :        ----}! xcsGetEnvironmentString (rc=xe
00000F13 16:18:56.381929   2512.1            :        ---}! xcsGetEnvironmentInteger (rc=xe
00000F14 16:18:56.381941   2512.1            :        ---{  xcsGetEnvironmentInteger
00000F15 16:18:56.381952   2512.1            :        ----{  xcsGetEnvironmentString
00000F16 16:18:56.381976   2512.1            :        xcsGetEnvironmentString[AMQ_CBM_LEN]
00000F17 16:18:56.381992   2512.1            :        ----}! xcsGetEnvironmentString (rc=xe
00000F18 16:18:56.382003   2512.1            :        ---}! xcsGetEnvironmentInteger (rc=xe
00000F19 16:18:56.382016   2512.1            :        --}  InitProcessInitialisation (rc=OK
00000F1A 16:18:56.383045   2512.1            :        --{  DLLMain
00000F1B 16:18:56.383059   2512.1            :        ---{  MCSInitCriticalSection
00000F1C 16:18:56.383068   2512.1            :        ---}  MCSInitCriticalSection (rc=OK)
```

Parent topic: »Tracing WebSphere MQ for Windows«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15310_

6.7.7.2. Tracing WebSphere MQ for UNIX systems

This is a brief introduction to the trace control commands you can use in Websphere MQ for UNIX.

In WebSphere® MQ for UNIX systems you enable or modify tracing using the strmqtrc control command, which is described in strmqtrc. To stop tracing, you use the endmqtrc control command, which is described in endmqtrc. On WebSphere MQ for Linux (x86 platform) systems you can alternatively use the WebSphere MQ Explorer to start and stop tracing, however, the function provided allows you to trace only everything, equivalent to using the commands strmqtrc -e and endmqtrc -e.

Trace output is unformatted; use the dspmqtrc control command to format trace output before viewing. For example, to format all trace files in the current directory use the following command:

```
dspmqtrc *.TRC
```

For more information on the control command dspmqtrc, see dspmqtrc.

»Selective component tracing on WebSphere MQ for UNIX systems«

Selective process tracing on WebSphere MQ for UNIX
This introduces the concept of selective process tracing on Websphere MQ for UNIX.

»Example trace data for WebSphere MQ for UNIX systems«

Parent topic: »Tracing«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15320_

6.7.7.2.1. Selective component tracing on WebSphere® MQ for UNIX

systems

Use the -t and -x options to control the amount of trace detail to record. By default, all trace points are enabled. The -x option enables you to specify the points you do not want to trace. So if, for example, you want to trace, for queue manager QM1, only output data associated with using Secure Sockets Layer (SSL) channel security, use:

```
strmqtrc -m QM1 -t ssl
```

For a full description of the trace command, see strmqtrc.
Parent topic: »Tracing WebSphere MQ for UNIX systems«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15330_

6.7.7.2.2. Selective process tracing on WebSphere® MQ for UNIX

This introduces the concept of selective process tracing on Websphere MQ for UNIX.

Use the -p option of the strmqtrc command control to restrict trace generation to specified named processes. For example, to trace all threads that result from any executing process called amqxxx.exe, use the following command:

```
strmqtrc -p amqxxx.exe
```

For a full description of the trace command, see strmqtrc.
Parent topic: »Tracing WebSphere MQ for UNIX systems«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15335_

6.7.7.2.3. Example trace data for WebSphere MQ for UNIX systems

»Figure 1 shows an extract from a WebSphere® MQ for HP-UX trace:
Figure 1. Sample WebSphere MQ for HP-UX trace

```
    Timestamp          Process.Thread   Trace Ident   Trace Data
    ============================================================
     11:36:30.398557     26693.1             :             Header.v02:7.0:HP-UX B.11.23:64:-1:1:G
     11:36:30.398645     26693.1             :             Version : 7.0.0.0     Level : p000-L09
     11:36:30.398681     26693.1             :             UTC    Date : 05/15/09   Time : 10:36:3
     11:36:30.398705     26693.1             :             Local Date : 05/15/09   Time : 11:36:3
     11:36:30.398751     26693.1             :             PID : 26693 Process : dltmqm_nd (64-b:
     11:36:30.398759     26693.1             :             Host : nemo01
     11:36:30.398765     26693.1             :             Operating System : HP-UX B.11.23
     11:36:30.398772     26693.1             :             Product Long Name : WebSphere MQ for
     11:36:30.398779     26693.1             :             ------------------------------
     11:36:30.398786     26693.1             :             xtrNullFd: 4, xihTraceFileNum: 5
     11:36:30.398820     26693.1             :             Thread stack
     11:36:30.398847     26693.1             :             -> InitProcessInitialisation
     11:36:30.398854     26693.1             :       {   InitProcessInitialisation
     11:36:30.398860     26693.1             :       -{  xcsIsEnvironment
     11:36:30.398911     26693.1             :             xcsIsEnvironment[AMQ_NO_CS_RELOAD] = F
     11:36:30.398918     26693.1             :       -}  xcsIsEnvironment rc=OK
     11:36:30.398980     26693.1             :       -{  xcsLoadFunction
     11:36:30.398997     26693.1             :             LibName(libmqmcs_r.sl) LoadType(209720
     11:36:30.399015     26693.1             :             General, comms, CS, OAM, or WAS
     11:36:30.399030     26693.1             :       --{  xcsQueryValueForSubpool
     11:36:30.399055     26693.1             :       --}  xcsQueryValueForSubpool rc=OK
     11:36:30.399138     26693.1             :             FullPathLibName(/opt/mqm/lib64/libmqmc
     11:36:30.399145     26693.1             :       --{  xcsGetMem
     11:36:30.399156     26693.1             :             component:24 function:176 length:2088
     11:36:30.399162     26693.1             :       --}  xcsGetMem rc=OK
     11:36:30.399172     26693.1             :             Handle(0000000000000000) Function(0000
     11:36:30.399178     26693.1             :       -}  xcsLoadFunction rc=OK
```

»Figure 2 shows an extract from a WebSphere MQ for Solaris trace:
Figure 2. Sample WebSphere MQ for Solaris trace

```
Timestamp          Process.Thread   Trace Ident   Trace Data
============================================================
 11:48:57.905466      7078.1             :             Header.v02:7.0:SunOS 5.9:64:-1:1:GMT
 11:48:57.905625      7078.1             :             Version : 7.0.0.0     Level : p000-L09
 11:48:57.905770      7078.1             :             UTC    Date : 05/15/09  Time : 10:48:5
 11:48:57.905816      7078.1             :             Local Date : 05/15/09  Time : 11:48:5
 11:48:57.906104      7078.1             :             PID : 7078 Process : dltmqm_nd (64-bi
 11:48:57.906129      7078.1             :             Host : computer.v6.hursley.ibm.com
 11:48:57.906148      7078.1             :             Operating System : SunOS 5.9
 11:48:57.906167      7078.1             :             Product Long Name : WebSphere MQ for
 11:48:57.906184      7078.1             :             --------------------------------
 11:48:57.906203      7078.1             :             xtrNullFd: 4, xihTraceFileNum: 5
 11:48:57.906276      7078.1             :             Thread stack
 11:48:57.906353      7078.1             :       {  xcsInitialize
 11:48:57.906385      7078.1             :       -{  InitPrivateServices
 11:48:57.906439      7078.1             :       --{  xcsGetEnvironmentString
 11:48:57.906566      7078.1             :             xcsGetEnvironmentString[MQS_ACTION_ON_
 11:48:57.906608      7078.1             :       --}! xcsGetEnvironmentString rc=xecE_E_ENV_
 11:48:57.906709      7078.1             :       --{  xcsIsEnvironment
 11:48:57.906738      7078.1             :             xcsIsEnvironment[AMQ_SIGCHLD_SIGACTION
 11:48:57.906755      7078.1             :       --}  xcsIsEnvironment rc=OK
 11:48:57.906771      7078.1             :             AMQ_SIGCHLD_SIGACTION is not set
 11:48:57.906835      7078.1             :       --{  xcsIsEnvironment
 11:48:57.906862      7078.1             :             xcsIsEnvironment[MQS_NO_SYNC_SIGNAL_HA
 11:48:57.906878      7078.1             :       --}  xcsIsEnvironment rc=OK
 11:48:57.907000      7078.1             :             FPE Handler installed, New=7e0b0f38, (
 11:48:57.907035      7078.1             :             SEGV Handler installed, New=7e0b0f38,
 11:48:57.907063      7078.1             :             BUS Handler installed, New=7e0b0f38, (
 11:48:57.907091      7078.1             :             ILL Handler installed, New=7e0b0f38, (
 11:48:57.907109      7078.1             :             Synchronous Signal Handling Activated
```

Figure 3 shows an extract from a WebSphere MQ for Linux trace:

Figure 3. Sample WebSphere MQ for Linux trace

```
Timestamp          Process.Thread   Trace Ident   Trace Data
============================================================
 11:02:23.643879      1239.1             :             Header.v02:7.0:Linux 2.6.5-7.276-smp:3
 11:02:23.643970      1239.1             :             Version : 7.0.0.0     Level : p000-L09
 11:02:23.644025      1239.1             :             UTC    Date : 05/15/09  Time : 10:02:2
 11:02:23.644054      1239.1             :             Local Date : 05/15/09  Time : 11:02:2
 11:02:23.644308      1239.1             :             PID : 1239 Process : dltmqm (32-bit)
 11:02:23.644324      1239.1             :             Host : hal1
 11:02:23.644334      1239.1             :             Operating System : Linux 2.6.5-7.276-s
 11:02:23.644344      1239.1             :             Product Long Name : WebSphere MQ for
 11:02:23.644353      1239.1             :             --------------------------------
 11:02:23.644363      1239.1             :             xtrNullFd: 3, xihTraceFileNum: 4
 11:02:23.644394      1239.1             :             Thread stack
 11:02:23.644412      1239.1             :             -> InitProcessInitialisation
 11:02:23.644427      1239.1             :       {  InitProcessInitialisation
 11:02:23.644439      1239.1             :       -{  xcsIsEnvironment
 11:02:23.644469      1239.1             :             xcsIsEnvironment[AMQ_NO_CS_RELOAD] = F
 11:02:23.644485      1239.1             :       -}  xcsIsEnvironment rc=OK
 11:02:23.644504      1239.1             :       -{  xcsLoadFunction
 11:02:23.644519      1239.1             :             LibName(libmqmcs_r.so) LoadType(209720
 11:02:23.644537      1239.1             :             General, comms, CS, OAM, or WAS
 11:02:23.644558      1239.1             :       --{  xcsQueryValueForSubpool
 11:02:23.644579      1239.1             :       --}  xcsQueryValueForSubpool rc=OK
 11:02:23.644641      1239.1             :             FullPathLibName(/opt/mqm/lib/libmqmcs_
 11:02:23.644652      1239.1             :       --{  xcsGetMem
 11:02:23.644675      1239.1             :             component:24 function:176 length:8212
 11:02:23.644685      1239.1             :       --}  xcsGetMem rc=OK
 11:02:23.644722      1239.1             :             Handle((nil)) Function((nil)) FullPath
 11:02:23.644732      1239.1             :       -}  xcsLoadFunction rc=OK
 11:02:23.644753      1239.1             :             SystemPageSize is 4096.
```

Figure 4 shows an extract from a WebSphere MQ for AIX® trace:

Figure 4. Sample WebSphere MQ for AIX trace

```
Timestamp          Process.Thread   Trace Ident   Trace Data
============================================================
 12:06:32.904335    622742.1           :             Header.v02:7.0:AIX 5.3:64:-1:1:GMT
 12:06:32.904427    622742.1           :             Version : 7.0.0.0     Level : p000-L09
 12:06:32.904540    622742.1           :             UTC    Date : 05/15/09  Time : 11:06:3
 12:06:32.904594    622742.1           :             Local Date : 05/15/09  Time : 12:06:3
 12:06:32.904697    622742.1           :             PID : 622742 Process : dltmqm_nd (64-b
```

```
12:06:32.904728    622742.1           :            Host : dynamo
12:06:32.904755    622742.1           :            Operating System : AIX 5.3
12:06:32.904781    622742.1           :            Product Long Name : WebSphere MQ for
12:06:32.904806    622742.1           :            --------------------------------
12:06:32.904832    622742.1           :            xtrNullFd: 3, xihTraceFileNum: 5
12:06:32.904916    622742.1           :            Data: 0x00000000
12:06:32.904952    622742.1           :            Thread stack
12:06:32.904982    622742.1           :            -> InitProcessInitialisation
12:06:32.905007    622742.1           :       {  InitProcessInitialisation
12:06:32.905033    622742.1           :       -{  xcsIsEnvironment
12:06:32.905062    622742.1           :             xcsIsEnvironment[AMQ_NO_CS_RELOAD] = F
12:06:32.905088    622742.1           :       -}  xcsIsEnvironment rc=OK
12:06:32.905117    622742.1           :       -{  xcsLoadFunction
12:06:32.905145    622742.1           :             LibName(libmqmcs_r.a(shr.o)) LoadType(
12:06:32.905178    622742.1           :             General, comms, CS, OAM, or WAS
12:06:32.905204    622742.1           :       --{  xcsQueryValueForSubpool
12:06:32.905282    622742.1           :       --}  xcsQueryValueForSubpool rc=OK
12:06:32.905504    622742.1           :             FullPathLibName(/usr/mqm/lib64/libmqmc
12:06:32.905540    622742.1           :       --{  xcsGetMem
12:06:32.905575    622742.1           :             component:24 function:176 length:2088
12:06:32.905601    622742.1           :       --}  xcsGetMem rc=OK
12:06:32.905638    622742.1           :             Handle(0) Function(0) FullPathLibName(
12:06:32.905665    622742.1           :       -}  xcsLoadFunction rc=OK
```

Parent topic: »Tracing WebSphere MQ for UNIX systems«

This build: January 26, 2011 11:55:21

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15340_

6.7.7.3. Trace files

Introduction to trace files for UNIX and Linux systems; their file paths and their file names.

»All trace files are created in the directory /var/mqm/trace.«

Note: »You can accommodate the production of large trace files by mounting a temporary file system over the directory that contains your trace files. Alternatively, rename the trace directory and create the symbolic link /var/mqm/trace to a different directory. «

Trace files are named AMQ*ppppp.qq*.TRC, where:

ppppp	Is the ID of the process reporting the error.
qq	Starts at 0. If the full file name exists, this value is incremented by one until a unique trace file name is found. A trace file name can already exist if a process is reused.

Note:

1. The process identifier can contain fewer, or more, digits than shown in the example.
2. There is one trace file for each process running as part of the entity being traced.

To format or view a trace file, you must be either the creator of the trace file, or a member of the mqm group.

Parent topic: »Tracing«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15350_

6.7.7.4. Tracing Secure Sockets Layer (SSL) iKeyman and iKeycmd functions

To request iKeyman tracing, execute the iKeyman command for your platform with the following -D flags.

For UNIX:

```
gsk7ikm -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

For Windows:

```
strmqikm -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

To request iKeycmd tracing, run the iKeycmd command for your platform with the following -D flags.

For UNIX:

```
gsk7cmd -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

For Windows:

```
runmqckm -Dkeyman.debug=true -Dkeyman.jnitracing=ON
```

iKeyman and iKeycmd write three trace files to the directory from which you start them, so consider starting iKeyman or iKeycmd from the trace directory to which the runtime SSL trace is written: /var/mqm/trace on UNIX and *<mqmtop>*/trace on Windows. The trace files that iKeyman and iKeycmd generate are:

ikmgdbg.log

Java related trace

ikmjdbg.log

JNI related trace

ikmcdbg.log

C related trace

These trace files are binary, so they must be transferred in binary transfer mode when they are transferred from system to system using FTP. The trace files are typically approximately 1 MB each.

On UNIX and Windows systems, you can independently request trace information for iKeyman, iKeycmd, the runtime SSL functions, or a combination of these.

The runtime SSL trace files have the names AMQ.SSL.TRC and AMQ.SSL.TRC.1. You cannot format any of the SSL trace files; send them unchanged to IBM® support. The SSL trace files are binary files and, if they are transferred to IBM support via FTP, they must be transferred in binary transfer mode.

Parent topic: »Tracing«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15360_

6.7.7.5. Tracing with the AIX system trace

In addition to the WebSphere® MQ trace, WebSphere MQ for AIX® users can use the standard AIX system trace. AIX system tracing is a two-step process:

1. Gathering the data
2. Formatting the results

WebSphere MQ uses two trace hook identifiers:

X'30D'

This event is recorded by WebSphere MQ on entry to or exit from a subroutine.

X'30E'

This event is recorded by WebSphere MQ to trace data such as that being sent or received across a communications network.

Trace provides detailed execution tracing to help you to analyze problems. IBM® service support personnel might ask for a problem to be re-created with trace enabled. The files produced by trace can be very large so it is important to qualify a trace, where possible. For example, you can optionally qualify a trace by time and by component.

There are two ways to run trace:

1. Interactively.

The following sequence of commands runs an interactive trace on the program myprog and ends the trace.

```
trace -j30D,30E -o trace.file
->!myprog
->q
```

2. Asynchronously.
   The following sequence of commands runs an asynchronous trace on the program myprog and ends the trace.

```
trace -a -j30D,30E -o trace.file
myprog
trcstop
```

You can format the trace file with the command:

```
trcrpt -t /usr/mqm/lib/amqtrc.fmt trace.file > report.file
```

report.file is the name of the file where you want to put the formatted trace output.

Note: All WebSphere MQ activity on the machine is traced while the trace is active.

»Selective component tracing on WebSphere MQ for AIX«

»An example of WebSphere MQ for AIX trace data«

Parent topic: »Tracing«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15370_

6.7.7.5.1. Selective component tracing on WebSphere® MQ for AIX®

Use the environment variable MQS_TRACE_OPTIONS to activate the high detail and parameter tracing functions individually. Because it enables tracing to be active without these functions, you can use it to reduce the impact on performance and trace size when you are trying to reproduce a problem with tracing switched on.

Only set the environment variable MQS_TRACE_OPTIONS if you have been instructed to do so by your service personnel.

Typically MQS_TRACE_OPTIONS must be set in the process that starts the queue manager, and before the queue manager is started, or it is not recognized. Set MQS_TRACE_OPTIONS before tracing starts. If it is set after tracing starts it is not recognized.

»SSL trace«

Parent topic: »Tracing with the AIX system trace«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15380_

6.7.7.5.1.1. SSL trace

If you request SSL trace, note the following:

- SSL trace is written to the directory /var/mqm/trace.
- The SSL trace files are AMQ.SSL.TRC and AMQ.SSL.TRC.1.
- You cannot format SSL trace files; send them unchanged to IBM® support.
- »The SSL trace files are binary files and, if they are transferred to IBM support via FTP, they must be transferred in binary transfer mode.«

Parent topic: »Selective component tracing on WebSphere MQ for AIX«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15390_

6.7.7.5.2. An example of WebSphere® MQ for AIX trace data

The following example is an extract of an AIX® trace:

Figure 1. Sample WebSphere MQ for AIX trace

```
   ID      ELAPSED_SEC     DELTA_MSEC    APPL     SYSCALL KERNEL  INTERRUPT

   30D     0.000000000       0.000000    MQS FNC Exit!..... 23298.1 ziiSendReceiveAgent
                                                                    rc=00000814
   30D     0.000009512       0.009512    MQS FNC Entry...... 23298.1 zcpDeleteMessage
   30D     0.000011869       0.002357    MQS FNC Exit....... 23298.1 zcpDeleteMessage
                                                                    rc=00000000
   30D     0.000014196       0.002327    MQS FNC Exit!..... 23298.1 ziiSPIInq1
                                                                    rc=00000814
   30D     0.000016727       0.002531    MQS FNC Exit!.... 23298.1 lpiSPIInq1
                                                                    rc=00000814
   30D     0.000019847       0.003120    MQS FNC Entry.... 23298.1 lpiSPIInq1
   30D     0.000022465       0.002618    MQS FNC Entry..... 23298.1 zstVerifyPCD
   30D     0.000024792       0.002327    MQS FNC Exit...... 23298.1 zstVerifyPCD
                                                                    rc=00000000
   30D     0.000027505       0.002713    MQS FNC Entry..... 23298.1 xcsCheckPointer
   30D     0.000032436       0.004931    MQS FNC Exit...... 23298.1 xcsCheckPointer
                                                                    rc=00000000
   30D     0.000034923       0.002487    MQS FNC Entry..... 23298.1 xcsCheckPointer
   30D     0.000039716       0.004793    MQS FNC Exit...... 23298.1 xcsCheckPointer
                                                                    rc=00000000
   30D     0.000042218       0.002502    MQS FNC Entry..... 23298.1 xcsCheckPointer
   30D     0.000046982       0.004764    MQS FNC Exit...... 23298.1 xcsCheckPointer
                                                                    rc=00000000
   30D     0.000049593       0.002611    MQS FNC Entry..... 23298.1 ziiSPIInq1
   30D     0.000052116       0.002523    MQS FNC Entry.... 23298.1 ziiCreateIPCCMessage
   30D     0.000054611       0.002495    MQS FNC Entry..... 23298.1 zcpCreateMessage
   30E     0.000059062       0.004451    Terminus(0) RequestedSize(236)
   30D     0.000061549       0.002487    MQS FNC Exit..... 23298.1 zcpCreateMessage
                                                                    rc=00000000
   30D     0.000063884       0.002335    MQS FNC Exit..... 23298.1 ziiCreateIPCCMessage
                                                                    rc=00000000
```

Parent topic: »Tracing with the AIX system trace«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15400_

6.7.8. First-failure support technology (FFST)

This section describes the role of first-failure support technology (FFST™) for WebSphere® MQ.

»FFST: WebSphere MQ for Windows«
Describes the contents of the First Failure Support Technology (FFST) file.

»FFST: WebSphere MQ for UNIX systems«
Describes the contents of the First Failure Support Technology (FFST) file.

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:22

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa15410_

6.7.8.1. FFST: WebSphere MQ for Windows

Describes the contents of the First Failure Support Technology (FFST™) file.

In WebSphere® MQ for Windows, FFST information is recorded in a file in the c:\Program Files\IBM\WebSphere MQ\errors directory.

An FFST file contains one or more records. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records typically indicate either a configuration problem with the system or a WebSphere MQ internal error.

FFST files are named AMQ*nnnnn.mm*.FDC, where:

nnnnn	Is the ID of the process reporting the error
mm	Starts at 0. If the full file name already exists, this value is incremented by one until a unique FFST file name is found. An FFST file name can already exist if a process is reused.

An instance of a process will write all FFST information to the same FFST file. If multiple errors occur during a single execution of the process, an FFST file can contain many records.

When a process writes an FFST record it also sends a record to the Event Log. The record contains the name of the FFST file to assist in automatic problem tracking. The Event log entry is made at the application level.

A typical FFST log is shown in Figure 1.

Figure 1. Sample WebSphere MQ for Windows First Failure Symptom Report

```
+------------------------------------------------------------------------+
| WebSphere MQ First Failure Symptom Report                              |
| =========================================                              |
|                                                                        |
| Date/Time           :- Mon January 28 2008 21:59:06 GMT                |
| UTC Time/Zone       :- 1201539869.892015 0 GMT                         |
| Host Name           :- 99VXY09 (Windows XP Build 2600: Service Pack 1) |
| PIDS                :- 5724H7200                                       |
| LVLS                :- 7.0.0.0                                         |
| Product Long Name :- WebSphere MQ for Windows                          |
| Vendor              :- IBM                                             |
| Probe Id            :- HL010004                                        |
| Application Name  :- MQM                                               |
| Component           :- hlgReserveLogSpace                              |
| SCCS Info           :- lib/logger/amqhlge0.c, 1.26                     |
| Line Number         :- 246                                             |
| Build Date          :- Jan 25 2008                                     |
| CMVC level          :- p000-L050202                                    |
| Build Type          :- IKAP - (Production)                             |
| UserID              :- IBM_User                                        |
| Process Name        :- C:\Program Files\IBM\WebSphere MQ\bin\amqzlaa0.exe |
| Process             :- 00003456                                        |
| Thread              :- 00000030                                        |
| QueueManager        :- qmgr2                                           |
| ConnId(1) IPCC      :- 162                                             |
| ConnId(2) QM        :- 45                                              |
| Major Errorcode     :- hrcE_LOG_FULL                                   |
| Minor Errorcode     :- OK                                              |
| Probe Type          :- MSGAMQ6709                                      |
| Probe Severity      :- 2                                               |
| Probe Description :- AMQ6709: The log for the Queue manager is full.   |
| FDCSequenceNumber :- 0                                                 |
+------------------------------------------------------------------------+

MQM Function Stack
zlaMainThread
zlaProcessMessage
zlaProcessMQIRequest
zlaMQPUT
zsqMQPUT
kpiMQPUT
kqiPutIt
kqiPutMsgSegments
apiPutMessage
aqmPutMessage
```

```
aqhPutMessage
aqqWriteMsg
aqqWriteMsgData
aqlReservePutSpace
almReserveSpace
hlgReserveLogSpace
xcsFFST

MQM Trace History
-------------} hlgReserveLogSpace rc=hrcW_LOG_GETTING_VERY_FULL
-------------{ xllLongLockRequest
-------------} xllLongLockRequest rc=OK

...
```

The Function Stack and Trace History are used by IBM® to assist in problem determination. In many cases there is little that the system administrator can do when an FFST record is generated, apart from raising problems through the IBM Support Center.

In certain circumstances a small dump file can be generated in addition to an FFST file and placed in the c:\Program Files\IBM\WebSphere MQ\errors directory. A dump file will have the same name as the FFST file, in the form AMQnnnnn.mm.dmp. These files can be used by IBM to assist in problem determination.

Parent topic: »First-failure support technology (FFST)«

This build: January 26, 2011 11:55:22

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15420_

6.7.8.2. FFST: WebSphere MQ for UNIX systems

Describes the contents of the First Failure Support Technology (FFST™) file.

For WebSphere® MQ for UNIX systems, FFST information is recorded in a file in the /var/mqm/errors directory.

An FFST file contains one or more records. Each FFST record contains information about an error that is normally severe, and possibly unrecoverable. These records indicate either a configuration problem with the system or a WebSphere MQ internal error.

FFST files are named AMQ*nnnnn.mm*.FDC, where:

nnnnn	Is the ID of the process reporting the error
mm	Starts at 0. If the full file name already exists, this value is incremented by one until a unique FFST file name is found. An FFST file name can already exist if a process is reused.

An instance of a process will write all FFST information to the same FFST file. If multiple errors occur during a single execution of the process, an FFST file can contain many records.

In order to read the contents of a FFST file, you must be either the creator of the file, or a member of the mqm group.

When a process writes an FFST record, it also sends a record to syslog. The record contains the name of the FFST file to assist in automatic problem tracking. The syslog entry is made at the *user.error* level. See the operating-system documentation about syslog.conf for information about configuring this.

Some typical FFST data is shown in Figure 1.

Figure 1. FFST report for WebSphere MQ for UNIX systems

```
+-------------------------------------------------------------------------------+
|                                                                               |
| WebSphere MQ First Failure Symptom Report                                     |
| =========================================                                     |
|                                                                               |
| Date/Time           :- Mon January 28 2008 21:59:06 GMT                       |
| UTC Time/Zone       :- 1201539869.892015 0 GMT                                |
| Host Name           :- mqperfh2 (HP-UX B.11.23)                               |
| PIDS                :- 5724H7202                                              |
| LVLS                :- 7.0.0.0                                                |
```

```
| Product Long Name :- WebSphere MQ for HP-UX                                  |
| Vendor             :- IBM                                                      |
| Probe Id           :- XC034255                                                 |
| Application Name   :- MQM                                                      |
| Component          :- xcsWaitEventSem                                          |
| SCCS Info          :- lib/cs/unix/amqxerrx.c, 1.204                            |
| Line Number        :- 6262                                                     |
| Build Date         :- Jan 25 2008                                              |
| CMVC level         :- p000-L050203                                             |
| Build Type         :- IKAP - (Production)                                      |
| UserID             :- 00000106 (mqperf)                                        |
| Program Name       :- amqzmuc0                                                 |
| Addressing mode    :- 64-bit                                                   |
| Process            :- 15497                                                    |
| Thread             :- 1                                                        |
| QueueManager       :- CSIM                                                     |
| ConnId(2) QM       :- 4                                                        |
| Major Errorcode    :- OK                                                       |
| Minor Errorcode    :- OK                                                       |
| Probe Type         :- INCORROUT                                                |
| Probe Severity     :- 4                                                        |
| Probe Description :- AMQ6109: An internal WebSphere MQ error has occurred.   |
| FDCSequenceNumber :- 0                                                         |
|                                                                              |
+------------------------------------------------------------------------------+

MQM Function Stack
amqzmuc0
xcsWaitEventSem
xcsFFST

MQM Trace History
Data: 0x00003c87
--} xcsCheckProcess rc=OK
--{ xcsRequestMutexSem
--} xcsRequestMutexSem rc=OK

...
```

The Function Stack and Trace History are used by IBM® to assist in problem determination. In many cases there is little that the system administrator can do when an FFST report is generated, apart from raising problems through the IBM Support Center.

However, there are some problems that the system administrator might be able to solve. If the FFST shows *out of resource* or *out of space on device* descriptions when calling one of the IPC functions (for example, semop or shmget), it is likely that the relevant kernel parameter limit has been exceeded.

If the FFST report shows a problem with setitimer, it is likely that a change to the kernel timer parameters is needed.

To resolve these problems, increase the IPC limits, rebuild the kernel, and restart the machine. See one of the following for further information:

- WebSphere MQ for AIX Quick Beginnings
- WebSphere MQ for HP-UX Quick Beginnings
- WebSphere MQ for Linux Quick Beginnings
- WebSphere MQ for Solaris Quick Beginnings

Parent topic: First-failure support technology (FFST)

This build: January 26, 2011 11:55:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15430_

6.7.9. Problem determination with WebSphere® MQ clients

An MQI client application receives MQRC_* reason codes in the same way as non-client MQI applications. However, there are additional reason codes for error conditions associated with clients. For example:

- Remote machine not responding

- Communications line error
- Invalid machine address

The most common time for errors to occur is when an application issues an MQCONN and receives the response MQRC_Q_MQR_NOT_AVAILABLE. An error message, written to the client log file, explains the cause of the error. Messages might also be logged at the server, depending on the nature of the failure.

»Terminating clients«

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15440_

6.7.9.1. Terminating clients

Even though a client has terminated, the process at the server can still hold its queues open. Normally, this is only for a short time until the communications layer detects that the partner has gone.

Parent topic: »Problem determination with WebSphere MQ clients«

This build: January 26, 2011 11:55:23

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15450_

6.7.10. Java diagnostics

For Java components of WebSphere® MQ, for example the WebSphere MQ Explorer and the Java implementation of WebSphere MQ Transport for SOAP, diagnostic information is output using the standard WebSphere MQ diagnostic facilities or by Java diagnostic classes.

Diagnostic information in this context consists of trace, first-failure data capture (FFDC) and error messages.

You can choose to have this information produced using WebSphere MQ facilities or the facilities of WebSphere MQ classes for Java or WebSphere MQ classes for JMS, as appropriate. You should generally use the WebSphere MQ diagnostic facilities if they are available on the local system.

You might want to use the Java diagnostics in the following circumstances:

- On a system on which queue managers are available, if the queue manager is managed separately from the software you are running.
- To reduce performance impact of WebSphere MQ trace.

To request and configure diagnostic output, two system properties are used when starting a WebSphere MQ Java process:

- System property com.ibm.mq.commonservices specifies a standard Java property file, which contains a number of lines which are used to configure the diagnostic outputs. Each line of code in the file is free-format, and is terminated by a new line character.
- System property com.ibm.mq.commonservices.diagid associates trace and FFDC files with the process which created them.

For information about tracing

»Using com.ibm.mq.commonservices«

»Java trace and FFDC files«

Parent topic: »Problem determination«

This build: January 26, 2011 11:55:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15460_

6.7.10.1. Using com.ibm.mq.commonservices

The com.ibm.mq.commonservices properties file contains the following entries relating to the output of diagnostics from the Java components of WebSphere® MQ. Note that case is significant in all these entries.

Diagnostics.MQ=*enabled|disabled*

Are WebSphere MQ diagnostics to be used? If Diagnostics.MQ is enabled, diagnostic output is as for other WebSphere MQ components; trace output is controlled by the parameters in the strmqtrc and endmqtrc control commands, or the equivalent. The default is *enabled*.

Diagnostics.Java=*options*

Which components are traced using Java trace. Options are one or more of *explorer*, *soap*, and *wmqjavaclasses*, separated by commas, where "explorer" refers to the diagnostics from the WebSphere MQ Explorer, "soap" refers to the diagnostics from the running process within WebSphere MQ Transport for SOAP, and "wmqjavaclasses" refers to the diagnostics from the underlying WebSphere MQ Java classes. By default no components are traced.

Diagnostics.Java.Trace.Detail=*high|medium|low*

Detail level for Java trace. The *high* and *medium* detail levels match those used in WebSphere MQ tracing but *low* is unique to Java trace. This property is ignored if Diagnostics.Java is not set. The default is *medium*.

Diagnostics.Java.Trace.Destination.File=*enabled|disabled*

Whether Java trace is written to a file. This property is ignored if Diagnostics.Java is not set. The default is *disabled*.

Diagnostics.Java.Trace.Destination.Console=*enabled|disabled*

Whether Java trace is written to the system console. This property is ignored if Diagnostics.Java is not set. The default is *disabled*.

Diagnostics.Java.Trace.Destination.Pathname=*dirname*

The directory to which Java trace is written. This property is ignored if Diagnostics.Java is not set or Diagnostics.Java.Trace.Destination.File=disabled. On UNIX systems, the default is /var/mqm/trace if it is present, otherwise the Java console (System.err). On Windows, the default is the system console.

Diagnostics.Java.FFDC.Destination.Pathname=*dirname*

The directory to which Java FFDC output is written. The default is the current working directory.

Diagnostics.Java.Errors.Destination.Filename=*filename*

The fully qualified file name to which Java error messages are written. The default is AMQJAVA.LOG in the current working directory.

An example of a com.ibm.mq.commonservices properties file is given in Figure 1. Lines beginning with the number sign (#) are treated as comments.

Figure 1. Sample com.ibm.mq.commonservices properties file

```
#
# Base WebSphere MQ diagnostics are disabled
#
Diagnostics.MQ=disabled
#
# Java diagnostics for WebSphere MQ Transport for SOAP
# and the WebSphere MQ Java Classes are both enabled
#
Diagnostics.Java=soap,wmqjavaclasses
#
# High detail Java trace
#
Diagnostics.Java.Trace.Detail=high
#
# Java trace is written to a file and not to the console.
#
Diagnostics.Java.Trace.Destination.File=enabled
Diagnostics.Java.Trace.Destination.Console=disabled
#
```

```
# Directory for Java trace file
#
Diagnostics.Java.Trace.Destination.Pathname=c:\\tracedir
#
# Directory for First Failure Data Capture
#
Diagnostics.Java.FFDC.Destination.Pathname=c:\\ffdcdir
#
# Directory for error logging
#
Diagnostics.Java.Errors.Destination.Filename=c:\\errorsdir\\SOAPERRORS.LOG
#
```

A sample properties file, WMQSoap_RAS.properties, is also supplied as part of the "Java messaging and SOAP transport" install option.

Parent topic: »Java diagnostics«

This build: January 26, 2011 11:55:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15470_

6.7.10.2. Java trace and FFDC files

When Java trace is generated for the WebSphere® MQ Explorer or for WebSphere MQ Transport for SOAP it is written to a file with a name of the format AMQ.*diagid*.*counter*.TRC where *diagid* is the value of the system property com.ibm.mq.commonservices.diagid associated with this Java process, as described earlier in this section, and *counter* is an integer greater than or equal to 0. All letters in the name are in upper case. This matches the naming convention used for normal WebSphere MQ trace.

If com.ibm.mq.commonservices.diagid is not specified, the value of *diagid* is the current time, in the format YYYYMMDDhhmmssmmm.

The WebSphere MQ Java classes trace file has a name based on the equivalent WebSphere MQ Explorer or SOAP Java trace file. The name differs in that it has the string .JC added before the .TRC string, giving a format of AMQ.*diagid*.*counter*.JC.TRC.

When Java FFDC is generated for the WebSphere MQ Explorer or for WebSphere MQ Transport for SOAP it is written to a file with a name of the format AMQ.*diagid*.*counter*.FDC where *diagid* and *counter* are as described for Java trace files.

Java error message output for the WebSphere MQ Explorer and for WebSphere MQ Transport for SOAP is written to the file specified by *Diagnostics.Java.Errors.Destination.Filename* for the appropriate Java process. The format of these files matches closely the format of the standard WebSphere MQ error logs.

When a process is writing trace information to a file, it appends to a single trace output file for the lifetime of the process. Similarly, a single FFDC output file is used for the lifetime of a process.

All trace output is in the UTF-8 character set.

Note that none of the above applies to output of FFDC data or error messages for the WebSphere MQ Java classes. This occurs as part of exception logging as detailed in WebSphere MQ Using Java. The WebSphere MQ Java class trace is also detailed in WebSphere MQ Using Java.

Parent topic: »Java diagnostics«

This build: January 26, 2011 11:55:23

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa15480_

6.8. WebSphere MQ and UNIX System V IPC resources

A queue manager uses some IPC resources. Use ipcs -a to find out what resources are being used.

This information applies to WebSphere® MQ running on UNIX systems only.

WebSphere MQ uses System V interprocess communication (IPC) resources (*semaphores* and *shared memory segments*) to store and pass data between system components. These resources are used by queue manager processes and applications that connect to the queue manager. WebSphere MQ clients do not use IPC resources, except for WebSphere MQ trace control. Use the UNIX command ipcs -a to get full information on the number and size of the IPC resources currently in use on the machine.

Clearing WebSphere MQ shared memory resources

Shared memory on AIX
If certain application types fail to connect because of an AIX® memory limitation, in most cases this can be resolved by setting the environment variable EXTSHM=ON.

Parent topic: Configuring and managing WebSphere MQ

This build: January 26, 2011 11:56:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22710_

6.8.1. Clearing WebSphere MQ shared memory resources

When a WebSphere® MQ queue manager is ended normally, the queue manager removes the majority of the IPC resources that it was using. A small number of IPC resources remain and this is as designed: some of the IPC resources are intended to persist between queue manager restarts. The number of IPC resources remaining varies to some extent, depending on the operating conditions.

There are some situations when a larger proportion of the IPC resources in use by a queue manager might persist after that queue manager has ended:

- If applications are connected to the queue manager when it stops (perhaps because the queue manager was shut down using endmqm -i or endmqm -p), the IPC resources used by these applications might not be released.
- If the queue manager ends abnormally (for example, if an operator issues the system kill command), some IPC resources might be left allocated after all queue manager processes have terminated.

In these cases, the IPC resources are not released back to the system until you restart (strmqm) or delete (dltmqm) the queue manager.

IPC resources allocated by WebSphere MQ are maintained automatically by the allocating queue managers. You are strongly recommended not to perform manual actions on or remove these IPC resources.

However, if it is necessary to remove IPC resources owned by mqm, follow these instructions. WebSphere MQ provides a utility to release the residual IPC resources allocated by a queue manager. This utility clears the internal queue manager state at the same time as it removes the corresponding IPC resource. Thus, this utility ensures that the queue manager state and IPC resource allocation are kept in step. To free residual IPC resources, follow these steps:

1. End the queue manager and all connecting applications.
2. Log on as user mqm.
3. Type the following:
   On Solaris, HP-UX, and Linux:

   ```
   /opt/mqm/bin/amqiclen -x -m QMGR
   ```

   On AIX®:

   ```
   /usr/mqm/bin/amqiclen -x -m QMGR
   ```

   This command does not report any status. However, if some WebSphere MQ-allocated resources could not be freed, the return code is nonzero.
4. Explicitly remove any remaining IPC resources that were created by user mqm.

Note: If a non-mqm application attempted to connect to WebSphere MQ before starting any queue managers, there might still be some WebSphere MQ IPC resources remaining even after following the above steps. These remaining resources were not created by user mqm and there

is no straightforward way to reliably recognize them. However, these resources are very small and are reused when WebSphere MQ is next restarted.

Parent topic: »WebSphere MQ and UNIX System V IPC resources«

This build: January 26, 2011 11:56:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22720_

6.8.2. Shared memory on AIX

If certain application types fail to connect because of an AIX® memory limitation, in most cases this can be resolved by setting the environment variable EXTSHM=ON.

Some 32-bit processes on AIX might encounter an operating system limitation that affects their ability to connect to WebSphere MQ queue managers. Every standard connection to WebSphere MQ uses shared memory, but unlike other UNIX platforms, AIX allows 32-bit processes to attach only 11 shared memory sets.

Most 32-bit processes will not encounter this limit, but applications with high memory requirements might fail to connect to WebSphere MQ with reason code 2102: MQRC_RESOURCE_PROBLEM. The following application types might see this error:

- Programs running in 32-bit Java virtual machines
- Programs using the large or very large memory models
- Programs connecting to many queue managers or databases
- Programs that attach to shared memory sets on their own

AIX offers an extended shared memory feature for 32-bit processes that allows them to attach more shared memory. To run an application with this feature, export the environment variable EXTSHM=ON before starting your queue managers and your program. The EXTSHM=ON feature prevents this error in most cases, but it is incompatible with programs that use the SHM_SIZE option of the shmctl function.

WebSphere MQ client applications and all 64-bit processes are unaffected by this limitation. They can connect to WebSphere MQ queue managers regardless of whether EXTSHM has been set.

Parent topic: »WebSphere MQ and UNIX System V IPC resources«

This build: January 26, 2011 11:56:13

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22730_

6.9. WebSphere MQ and UNIX Process Priority

Recommended ways to set process priority *nice* values.

This information applies to WebSphere® MQ running on UNIX systems only.

If you run a process in the background, that process can be given a higher *nice* value (and hence lower priority) by the invoking shell. This might have general WebSphere MQ performance implications. In highly-stressed situations, if there are many ready-to-run threads at a higher priority and some at a lower priority, operating system scheduling characteristics can deprive the lower priority threads of CPU time.

It is strongly recommended that independently started processes associated with queue managers, such as runmqlsr, have the same *nice* values as the queue manager they are associated with. Ensure the shell does not assign a higher *nice* value to these background processes. For example, in ksh, use the setting "set +o bgnice" to stop ksh from raising the *nice* value of background processes. You can verify the *nice* values of running processes by examining the *NI* column of a "ps -efl" listing.

It is also recommended that you start WebSphere MQ application processes with the same *nice* value as the queue manager. If they run with different *nice* values, an application thread might block a queue manager thread, or vice versa, causing performance to degrade.

Parent topic: »Configuring and managing WebSphere MQ«

This build: January 26, 2011 11:56:13

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22740_

7. WebSphere® MQ installable services and the API exit

Installable services and components
This section introduces the installable services and the functions and components associated with them. The interface to these functions is documented so that you, or software vendors, can supply components.

Authorization service

Name service

Installable services interface reference information
This collection of topics provides reference information for the installable services.

API exits

API exit reference information

Referencing connection definitions using a pre-connect exit from a repository
WebSphere MQ clients can be configured to look up a repository to obtain connection definitions using a pre-connect exit library.

Parent topic: System Administration Guide

This build: January 26, 2011 11:55:45

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16180_

7.1. Installable services and components

This section introduces the installable services and the functions and components associated with them. The interface to these functions is documented so that you, or software vendors, can supply components.

This section includes:

- Why installable services?
- Functions and components
- Initialization
- Configuring services and components
- Creating your own service component
- Using multiple service components

The installable services interface is described in Installable services interface reference information.

Why installable services?

Functions and components

Initialization

Configuring services and components

Creating your own service component

Using multiple service components

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16190_

7.1.1. Why installable services?

The main reasons for providing WebSphere® MQ installable services are:

- To provide you with the flexibility of choosing whether to use components provided by WebSphere MQ products, or replace or augment them with others.
- To allow vendors to participate, by providing components that might use new technologies, without making internal changes to WebSphere MQ products.
- To allow WebSphere MQ to exploit new technologies faster and cheaper, and so provide products earlier and at lower prices.

Installable services and *service components* are part of the WebSphere MQ product structure. At the center of this structure is the part of the queue manager that implements the function and rules associated with the Message Queue Interface (MQI). This central part requires a number of service functions, called *installable services*, in order to perform its work. The installable services are:

- Authorization service
- Name service

Each installable service is a related set of functions implemented using one or more *service components*. Each component is invoked using a properly-architected, publicly-available interface. This enables independent software vendors and other third parties to provide installable components to augment or replace those provided by the WebSphere MQ products. Table 1 summarizes the services and components that can be used.

Table 1. Installable service components summary

Installable service	Supplied component	Function	Requirements
Authorization service	Object Authority Manager (OAM)	Provides authorization checking on commands and MQI calls. Users can write their own component to augment or replace the OAM.	(Appropriate platform authorization facilities are assumed)
Name service	None	• User defined **Note:** Shared queues must have their *Scope* attribute set to CELL.	• A third-party or user-written name manager

Parent topic: Installable services and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16200_

7.1.2. Functions and components

Each service consists of a set of related functions. For example, the name service contains function for:

- Looking up a queue name and returning the name of the queue manager where the queue is defined
- Inserting a queue name into the service's directory
- Deleting a queue name from the service's directory

It also contains initialization and termination functions.

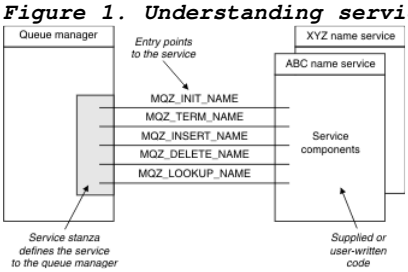
An installable service is provided by one or more service components. Each component can

perform some or all of the functions that are defined for that service. For example, in WebSphere® MQ for AIX®, the supplied authorization service component, the OAM, performs all the available functions. See Authorization service interface for more information. The component is also responsible for managing any underlying resources or software (for example, an LDAP directory) that it needs to implement the service. Configuration files provide a standard way of loading the component and determining the addresses of the functional routines that it provides.

Figure 1 shows how services and components are related:

- A service is defined to a queue manager by stanzas in a configuration file.
- Each service is supported by supplied code in the queue manager. Users cannot change this code and therefore cannot create their own services.
- Each service is implemented by one or more components; these can be supplied with the product or user-written. Multiple components for a service can be invoked, each supporting different facilities within the service.
- Entry points connect the service components to the supporting code in the queue manager.

Figure 1. Understanding services, components, and entry points



Entry-points

Return codes

Component data

Parent topic: Installable services and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16210_

7.1.2.1. Entry-points

Each service component is represented by a list of the entry-point addresses of the routines that support a particular installable service. The installable service defines the function to be performed by each routine.

The ordering of the service components when they are configured defines the order in which entry-points are called in an attempt to satisfy a request for the service.

In the supplied header file cmqzc.h, the supplied entry points to each service have an MQZID_ prefix.

If the services are present, the services are loaded in a predefined order. The following list shows the services, and the order in which they are initialized.

1. NameService
2. AuthorizationService
3. UserIdentifierService

The AuthorizationService is the only service that is configured by default. Configure the NameService and UserIdentifierService manually if you want to use them.

Services, and service components have a one-to-one or one-to-many mapping. Multiple service components can be defined for each service. On UNIX, the ServiceComponent stanza's Service value must match the Service stanza's Name value in the qm.ini file. On Windows, the ServiceComponent's Service registry key value must match the Name registry key value, and is defined as:

HKEY_LOCAL_MACHINE\SOFTWARE\IBM\MQSeries\CurrentVersion\Configuration\QueueManager*qmname*\
where *qmname* is the name of the queue manager.

For UNIX, service components are started in the order they are defined in the qm.ini file. On Windows, because the Windows registry is used, WebSphere® MQ issues a RegEnumKey call which returns the values in alphabetic order. Therefore, on Windows the services are called in alphabetic order, as they are defined in the registry.

The ordering of the ServiceComponent definitions is significant. This ordering dictates the order in which components are run for a given service. For example, the AuthorizationService on Windows is configured with the default OAM component named MQSeries.WindowsNT.auth.service. Additional components can be defined for this service in order to override the default OAM. Unless MQCACF_SERVICE_COMPONENT is specified, the first component encountered in alphabetic order is used to process the request, and the name for that component is used.

Parent topic: Functions and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16220_

7.1.2.2. Return codes

Service components provide return codes to the queue manager to report on a variety of conditions. They report the success or failure of the operation, and indicate whether the queue manager is to proceed to the next service component. A separate *Continuation* parameter carries this indication.

Parent topic: Functions and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16230_

7.1.2.3. Component data

A single service component might require data to be shared between its various functions. Installable services provide an optional data area to be passed on each invocation of a given service component. This data area is for the exclusive use of the service component. It is shared by all the invocations of a given function, even if they are made from different address spaces or processes. It is guaranteed to be addressable from the service component whenever it is called. You must declare the size of this area in the *ServiceComponent* stanza.

Parent topic: Functions and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16240_

7.1.3. Initialization

When the component initialization routine is invoked, it must call the queue manager MQZEP function for each entry-point supported by the component. MQZEP defines an entry-point to the service. All the undefined exit points are assumed to be NULL.

Primary initialization

Secondary initialization

Primary termination

Secondary termination

Parent topic: Installable services and components

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16250_

7.1.3.1. Primary initialization

A component is always invoked with this option once, before it is invoked in any other way.

Parent topic: Initialization

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16260_

7.1.3.2. Secondary initialization

A component can be invoked with this option on certain platforms. For example, it can be invoked once for each operating system process, thread, or task by which the service is accessed.

If secondary initialization is used:

- The component can be invoked more than once for secondary initialization. For each such call, a matching call for secondary termination is issued when the service is no longer needed.
  For naming services this is the MQZ_TERM_NAME call.
  For authorization services this is the MQZ_TERM_AUTHORITY call.
- The entry points must be re-specified (by calling MQZEP) each time the component is called for primary and secondary initialization.
- Only one copy of component data is used for the component; there is not a different copy for each secondary initialization.
- The component is not invoked for any other calls to the service (from the operating system process, thread, or task, as appropriate) before secondary initialization has been carried out.
- The component must set the *Version* parameter to the same value for primary and secondary initialization.

Parent topic: Initialization

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16270_

7.1.3.3. Primary termination

The component is always invoked with this option once, when it is no longer required. No further calls are made to this component.

Parent topic: Initialization

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16280_

7.1.3.4. Secondary termination

The component is invoked with this option, if it has been invoked for secondary initialization.

Parent topic: Initialization

This build: January 26, 2011 11:55:45

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16290_

7.1.4. Configuring services and components

Configure service components using the queue manager configuration files, except on Windows systems, where each queue manager has its own stanza in the Registry. Each service used must have a *Service* stanza, which defines the service to the queue manager.

For each component within a service, there must be a *ServiceComponent* stanza. This identifies the name and path of the module containing the code for that component.

The authorization service component, known as the Object Authority Manager (OAM), is supplied with the product. When you create a queue manager, the queue manager configuration file (or the Registry on Windows systems) is automatically updated to include the appropriate stanzas for the authorization service and for the default component (the OAM). For the other components, you must configure the queue manager configuration file manually.

The code for each service component is loaded into the queue manager when the queue manager is started, using dynamic binding, where this is supported on the platform.

Service stanza format

Service stanza format for Windows systems
On Windows systems, the *Service* stanza includes a SecurityPolicy attribute.

Service component stanza format

Parent topic: Installable services and components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16300_

7.1.4.1. Service stanza format

The format of the *Service* stanza is:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
```

where:

<service_name>

The name of the service. This is defined by the service.

<entries>

The number of entry-points defined for the service. This includes the initialization and termination entry points.

Parent topic: Configuring services and components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa16310_

7.1.4.2. Service stanza format for Windows systems

On Windows systems, the *Service* stanza includes a SecurityPolicy attribute.

The format of the stanza is:

```
Service:
  Name=<service_name>
  EntryPoints=<entries>
  SecurityPolicy=<policy>
```

where:

<service_name>

The name of the service. This is defined by the service.

<entries>

The number of entry-points defined for the service. This includes the initialization and termination entry points.

<policy>

NTSIDsRequired (the Windows Security Identifier), or Default. If you do not specify NTSIDsRequired, the Default value is used. This attribute is valid only if Name has a value of AuthorizationService.

See also Configuring authorization service stanzas: Windows systems.

Parent topic: Configuring services and components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16320_

7.1.4.3. Service component stanza format

The format of the *Service component* stanza is:

```
ServiceComponent:
  Service=<service_name>
  Name=<component_name>
  Module=<module_name>
  ComponentDataSize=<size>
```

where:

<service_name>

The name of the service. This must match the Name specified in a service stanza.

<component_name>

A descriptive name of the service component. This must be unique, and contain only the characters that are valid for the names of WebSphere® MQ objects (for example, queue names). This name occurs in operator messages generated by the service. We recommend that you use a name starting with a company trademark or similar distinguishing string.

<module_name>

The name of the module to contain the code for this component. Specify a full path name.

<size>

The size in bytes of the component data area passed to the component on each call. Specify zero if no component data is required.

These two stanzas can appear in any order and the stanza keys under them can also appear in any order. For either of these stanzas, all the stanza keys must be present. If a stanza key is duplicated, the last one is used.

At startup time, the queue manager processes each service component entry in the configuration file in turn. It then loads the specified component module, invoking the entry-point of the component (which must be the entry-point for initialization of the component), passing it a configuration handle.

Parent topic: Configuring services and components

This build: January 26, 2011 11:55:46

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16330_

7.1.5. Creating your own service component

To create your own service component:

- For all platforms, ensure that the header file cmqzc.h is included in your program.
- For UNIX systems: Create the shared library by compiling the program and linking it with the shared libraries libmqm* and libmqmzf*. (The threading suffixes and file extensions vary by platform.)
  Note: Because the agent can run in a threaded environment, you must build the OAM and Name Service to run in a threaded environment. This includes using the threaded versions of libmqm and libmqmzf.
- For Windows: Create a DLL by compiling the program, and linking it with the libraries MQM.LIB and MQMZF.LIB.
  See the WebSphere MQ Application Programming Guide for details of how to compile and link code for shared libraries.
- Add stanzas to the queue manager configuration file to define the service to the queue manager and to specify the location of the module. Refer to the individual chapters for each service, for more information.
- Stop and restart the queue manager to activate the component.

Parent topic: Installable services and components

This build: January 26, 2011 11:55:46

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16340_

7.1.6. Using multiple service components

You can install more than one component for a given service. This allows components to provide only partial implementations of the service, and to rely on other components to provide the remaining functions.

Example of using multiple components

Omitting entry points when using multiple components

Example of entry points used with multiple components

Parent topic: Installable services and components

This build: January 26, 2011 11:55:46

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16350_

7.1.6.1. Example of using multiple components

Suppose you create two a name services components called ABC_name_serv and XYZ_name_serv.

ABC_name_serv

This component supports inserting a name in, or deleting a name from, the service directory, but does not support looking up a queue name.

XYZ_name_serv

This component supports looking up a queue name, but does not support inserting a name in, or deleting a name from, the service directory.

What the components do

How the component is used

Parent topic: Using multiple service components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16360_

7.1.6.1.1. What the components do

Component ABC_name_serv holds a database of queue names, and uses two simple algorithms to either insert, or delete, a name from the service directory.

Component XYZ_name_serv uses a simple algorithm that returns a fixed queue-manager name for any queue name with which it is invoked. It does not hold a database of queue names, and therefore does not support the insert and delete functions.

Parent topic: Example of using multiple components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16370_

7.1.6.1.2. How the component is used

The components are installed on the same queue manager. The *ServiceComponent* stanzas are ordered so that component ABC_name_serv is invoked first. Any calls to insert or delete a queue in a component directory are handled by component ABC_name_serv; it is the only one that implements these functions. However, a lookup call that component ABC_name_serv cannot resolve is passed on to the lookup-only component, XYZ_name_serv. This component supplies a queue-manager name from its simple algorithm.

Parent topic: Example of using multiple components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16380_

7.1.6.2. Omitting entry points when using multiple components

If you decide to use multiple components to provide a service, you can design a service component that does not implement certain functions. The installable services framework places no restrictions on which you can omit. However, for specific installable services, omission of one or more functions might be logically inconsistent with the purpose of the service.

Parent topic: Using multiple service components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16390_

7.1.6.3. Example of entry points used with multiple components

Table 1 shows an example of the installable name service for which the two components have been installed. Each supports a different set of functions associated with this particular installable service. For insert function, the ABC component entry-point is invoked first.

Entry points that have not been defined to the service (using MQZEP) are assumed to be NULL. An entry-point for initialization is provided in the table, but this is not required because initialization is carried out by the main entry-point of the component.

When the queue manager has to use an installable service, it uses the entry-points defined for that service (the columns in Table 1). Taking each component in turn, the queue manager determines the address of the routine that implements the required function. It then calls the routine, if it exists. If the operation is successful, any results and status information are used by the queue manager.

Table 1. Example of entry-points for an installable service

Function number	ABC name service component	XYZ name service component
MQZID_INIT_NAME (Initialize)	ABC_initialize()	XYZ_initialize()
MQZID_TERM_NAME (Terminate)	ABC_terminate()	XYZ_terminate()
MQZID_INSERT_NAME (Insert)	ABC_Insert()	NULL
MQZID_DELETE_NAME (Delete)	ABC_Delete()	NULL
MQZID_LOOKUP_NAME (Lookup)	NULL	XYZ_Lookup()

If the routine does not exist, the queue manager repeats this process for the next component in the list. In addition, if the routine does exist but returns a code indicating that it could not perform the operation, the attempt continues with the next available component. Routines in service components might return a code that indicates that no further attempts to perform the operation should be made.

Parent topic: Using multiple service components

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16400_

7.2. Authorization service

The authorization service is an installable service that enables queue managers to invoke authorization facilities, for example, checking that a user ID has authority to open a queue.

This service is a component of the WebSphere® MQ security enabling interface (SEI), which is part of the WebSphere MQ framework.

This chapter discusses:

- Object authority manager (OAM)
- Authorization service on UNIX systems
- Authorization service on Windows systems
- Authorization service interface

Object authority manager (OAM)

Authorization service on UNIX systems

Authorization service on Windows systems

Authorization service interface

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16410_

7.2.1. Object authority manager (OAM)

The authorization service component supplied with the WebSphere® MQ products is called the Object Authority Manager (OAM). By default, the OAM is active and works with the control commands dspmqaut (display authority),dmpmqaut (dump authority), and setmqaut (set or reset authority).

The syntax of these commands and how to use them are described in The control commands.

The OAM works with the *entity* of a principal or group. These entities vary from platform to platform.

When an MQI request is made or a command is issued, the OAM checks the authorization of the entity associated with the operation to see whether it can:

- Perform the requested operation.
- Access the specified queue manager resources.

The authorization service enables you to augment or replace the authority checking provided for queue managers by writing your own authorization service component.

Defining the service to the operating system

Refreshing the OAM after changing a user's authorization

Migrating from MQSeries Version 5.1

Parent topic: Authorization service

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16420_

7.2.1.1. Defining the service to the operating system

The authorization service stanzas in the queue manager configuration file qm.ini (or Registry on Windows systems), define the authorization service to the queue manager. See Configuring services and components for information about the types of stanza.

Parent topic: Object authority manager (OAM)

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16430_

7.2.1.2. Refreshing the OAM after changing a user's authorization

In WebSphere® MQ, you can update the OAM's authorization group information immediately after changing a user's authorization group membership, reflecting changes made at the operating system level, without needing to stop and restart the queue manager.

Note: When you change authorizations with the setmqaut command, the OAM implements such changes immediately.

Queue managers store authorization data on a local queue called SYSTEM.AUTH.DATA.QUEUE. This data is managed by amqzfuma.exe.

Parent topic: Object authority manager (OAM)

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16440_

7.2.1.3. Migrating from MQSeries Version 5.1

All authorization data is migrated from the authorization files to the authorization queue the first time that you restart the queue manager after migrating from MQSeries® Version 5.1. If the OAM detects a missing file:

1. If the authorization applies to a single object, the OAM gives the mqm group (or Administrator on Windows systems) access to the object and continues with the migration. Message AMQ5528 is written to the queue manager's error log. Refer to WebSphere MQ Messages for more information about message AMQ5528.
2. If the authorization applies to a class of objects, the OAM stops the migration. The queue manager does not start until the file has been replaced.

Parent topic: Object authority manager (OAM)

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16450_

7.2.2. Authorization service on UNIX systems

On these platforms:

Principal

Is a UNIX system user ID, or an ID associated with an application program running on behalf of a user.

Group

Is a UNIX system-defined collection of principals.

Authorizations can be granted or revoked at the group level only. A request to grant or revoke a user's authority updates the primary group for that user.

Configuring authorization service stanzas: UNIX systems

Parent topic: Authorization service

This build: January 26, 2011 11:55:46

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16460_

7.2.2.1. Configuring authorization service stanzas: UNIX systems

On UNIX systems, each queue manager has its own queue manager configuration file.

For example, on AIX®, the default path and file name of the queue manager configuration file for queue manager QMNAME is /var/mqm/qmgrs/QMNAME/qm.ini.

The *Service* stanza and the *ServiceComponent* stanza for the default authorization component are added to qm.ini automatically, but can be overridden by mqsnoaut. Any other *ServiceComponent* stanzas must be added manually.

For example, the following stanzas in the queue manager configuration file define two authorization service components on WebSphere® MQ for AIX:

Figure 1. UNIX authorization service stanzas in qm.ini

```
Service:
   Name=AuthorizationService
   EntryPoints=13

ServiceComponent:
   Service=AuthorizationService
   Name=MQSeries.UNIX.auth.service
   Module=/usr/mqm/lib/amqzfu
   ComponentDataSize=0

ServiceComponent:
   Service=AuthorizationService
```

```
Name=user.defined.authorization.service
Module=/usr/bin/udas01
ComponentDataSize=96
```

The service component stanza (MQSeries.UNIX.auth.service) defines the default authorization service component, the OAM. If you remove this stanza and restart the queue manager, the OAM is disabled and no authorization checks are made.

Parent topic: Authorization service on UNIX systems

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16470_

7.2.3. Authorization service on Windows systems

On Windows systems:

Principal

Is a Windows user ID, or an ID associated with an application program running on behalf of a user.

Group

Is a Windows group.

Authorizations can be granted or revoked at the principal or group level.

Configuring authorization service stanzas: Windows systems
On WebSphere® MQ for Windows each queue manager has its own stanza in the registry.

Parent topic: Authorization service

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16480_

7.2.3.1. Configuring authorization service stanzas: Windows systems

On WebSphere® MQ for Windows each queue manager has its own stanza in the registry.

The *Service* stanza and the *ServiceComponent* stanza for the default authorization component are added to the Registry automatically, but can be overridden using mqsnoaut. Any other *ServiceComponent* stanzas must be added manually.

You can also add the SecurityPolicy attribute using the WebSphere MQ services. The SsecurityPolicy attribute applies only if the service specified on the *Service* stanza is the authorization service, that is, the default OAM. The SecurityPolicy attribute allows you to specify the security policy for each queue manager. The possible values are:

Default

Specify Default if you want the default security policy to take effect. If a Windows security identifier (NT SID) is not passed to the OAM for a particular user ID, an attempt is made to obtain the appropriate SID by searching the relevant security databases.

NTSIDsRequired

Requires that an NT SID is passed to the OAM when performing security checks.

For information about the Service stanza format, see Service stanza format for Windows systems. For more general information about security, see Setting up security on UNIX systems and Windows.

The service component stanza, MQSeries.WindowsNT.auth.service defines the default authorization service component, the OAM. If you remove this stanza and restart the queue manager, the OAM is disabled and no authorization checks are made.

Parent topic: Authorization service on Windows systems

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16490_

7.2.4. Authorization service interface

The authorization service provides the following entry points for use by the queue manager:

MQZ_AUTHENTICATE_USER

Authenticates a user ID and password, and can set identity context fields.

MQZ_CHECK_AUTHORITY

Checks whether an entity has authority to perform one or more operations on a specified object.

MQZ_COPY_ALL_AUTHORITY

Copies all the current authorizations that exist for a referenced object to another object.

MQZ_DELETE_AUTHORITY

Deletes all authorizations associated with a specified object.

MQZ_ENUMERATE_AUTHORITY_DATA

Retrieves all the authority data that matches the selection criteria specified.

MQZ_FREE_USER

Frees associated allocated resources.

MQZ_GET_AUTHORITY

Gets the authority that an entity has to access a specified object.

MQZ_GET_EXPLICIT_AUTHORITY

Gets either the authority that a named group has to access a specified object (but without the additional authority of the nobody group) or the authority that the primary group of the named principal has to access a specified object.

MQZ_INIT_AUTHORITY

Initializes authorization service component.

MQZ_INQUIRE

Queries the supported functionality of the authorization service.

MQZ_REFRESH_CACHE

Refresh all authorizations.

MQZ_SET_AUTHORITY

Sets the authority that an entity has to a specified object.

MQZ_TERM_AUTHORITY

Terminates authorization service component.

In addition, on WebSphere® MQ for Windows, the authorization service provides the following entry points for use by the queue manager:

- MQZ_CHECK_AUTHORITY_2
- MQZ_GET_AUTHORITY_2
- MQZ_GET_EXPLICIT_AUTHORITY_2
- MQZ_SET_AUTHORITY_2

These entry points support the use of the Windows Security Identifier (NT SID).

These names are defined as typedefs, in the header file cmqzc.h, which can be used to prototype the component functions.

The initialization function (MQZ_INIT_AUTHORITY) must be the main entry point for the component. The other functions are invoked through the entry point address that the initialization function has added into the component entry point vector.

See Creating your own service component for more information.

Parent topic: Authorization service

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16500_

7.3. Name service

The name service is an installable service that provides support to the queue manager for looking up the name of the queue manager that owns a specified queue. No other queue attributes can be retrieved from a name service.

The name service enables an application to open remote queues for output as if they were local queues. A name service is not invoked for objects other than queues.

Note: The remote queues *must* have their *Scope* attribute set to CELL.

When an application opens a queue, it looks for the name of the queue first in the queue manager's directory. If it does not find it there, it looks in as many name services as have been configured, until it finds one that recognizes the queue name. If none recognizes the name, the open fails.

The name service returns the owning queue manager for that queue. The queue manager then continues with the MQOPEN request as if the command had specified the queue and queue manager name in the original request.

The name service interface (NSI) is part of the WebSphere® MQ framework.

This chapter discusses:

- How the name service works

How the name service works

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16510_

7.3.1. How the name service works

If a queue definition specifies the *Scope* attribute as queue manager, that is, SCOPE(QMGR) in MQSC, the queue definition (along with all the queue attributes) is stored in the queue manager's directory only. This cannot be replaced by an installable service.

If a queue definition specifies the *Scope* attribute as cell, that is, SCOPE(CELL) in MQSC, the queue definition is again stored in the queue manager's directory, along with all the queue attributes. However, the queue and queue-manager name are also stored in a name service. If no service is available that can store this information, a queue with the *Scope* cell cannot be defined.

The directory in which the information is stored can be managed by the service, or the service can use an underlying service, for example, an LDAP directory, for this purpose. In either case, definitions stored in the directory must persist, even after the component and queue manager have terminated, until they are explicitly deleted.

Note:

1. To send a message to a remote host's local queue definition (with a scope of CELL) on a different queue manager within a naming directory cell, you need to define a channel.
2. You cannot get messages directly from the remote queue, even when it has a scope of CELL.
3. No remote queue definition is required when sending to a queue with a scope of CELL.
4. The naming service centrally defines the destination queue, although you still need a transmission queue to the destination queue manager and a pair of channel definitions. In addition, the transmission queue on the local system must have the same name as the queue manager owning the target queue, with the scope of cell, on the remote system.
   For example, if the remote queue manager has the name QM01, the transmission queue on the local system must also have the name QM01. See WebSphere MQ Intercommunication for further information.

Name service interface

Parent topic: Name service

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16520_

7.3.1.1. Name service interface

A name service provides the following entry points for use by the queue manager:

MQZ_INIT_NAME

Initialize the name service component.

MQZ_TERM_NAME

Terminate the name service component.

MQZ_LOOKUP_NAME

Look up the queue-manager name for the specified queue.

MQZ_INSERT_NAME

Insert an entry containing the owning queue-manager name for the specified queue into the directory used by the service.

MQZ_DELETE_NAME

Delete the entry for the specified queue from the directory used by the service.

If there is more than one name service configured:

- For lookup, the MQZ_LOOKUP_NAME function is invoked for each service in the list until the queue name is resolved (unless any component indicates that the search should stop).
- For insert, the MQZ_INSERT_NAME function is invoked for the first service in the list that supports this function.
- For delete, the MQZ_DELETE_NAME function is invoked for the first service in the list that supports this function.

Do not have more than one component that supports the insert and delete functions. However, a component that only supports lookup is feasible, and could be used, for example, as the last component in the list to resolve any name that is not known by any other name service component to a queue manager at which the name can be defined.

In the C programming language the names are defined as function data types using the typedef statement. These can be used to prototype the service functions, to ensure that the parameters are correct.

The header file that contains all the material specific to installable services is cmqzc.h for the C language.

Apart from the initialization function (MQZ_INIT_NAME), which must be the component's main entry point, functions are invoked by the entry point address that the initialization function has added, using the MQZEP call.

The following examples of UNIX configuration file stanzas for the name service specify a name service component provided by the (fictitious) ABC company.

Figure 1. Name service stanzas in qm.ini (for UNIX systems)

```
# Stanza for name service
 Service:
     Name=NameService
     EntryPoints=5

# Stanza for name service component, provided by ABC
 ServiceComponent:
     Service=NameService
     Name=ABC.Name.Service
     Module=/usr/lib/abcname
     ComponentDataSize=1024
```

Note: On Windows systems, name service stanza information is stored in the Registry.

Parent topic: How the name service works

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16530_

7.4. Installable services interface reference information

This collection of topics provides reference information for the installable services.

The functions and data types are listed in alphabetic order within the group for each service type.

How the functions are shown

MQZEP - Add component entry point
This function is invoked by a service component, during initialization, to add an entry point to the entry point vector for that service component.

MQZ_AUTHENTICATE_USER - Authenticate user
This function is provided by an MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to authenticate a user, or to set identity context fields. It is invoked when WebSphere® MQ's user application context is established.

MQZ_CHECK_AUTHORITY - Check authority
This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to check whether an entity has authority to perform a particular action, or actions, on a specified object.

MQZ_CHECK_AUTHORITY_2 - Check authority (extended)
This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to check whether an entity has authority to perform a particular action, or actions, on a specified object.

MQZ_COPY_ALL_AUTHORITY - Copy all authority
This function is provided by an authorization service component. It is invoked by the queue manager to copy all of the authorizations that are currently in force for a reference object to another object.

MQZ_DELETE_AUTHORITY - Delete authority
This function is provided by an authorization service component, and is invoked by the queue manager to delete all of the authorizations associated with the specified object.

MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data
This function is provided by an MQZAS_VERSION_4 authorization service component, and is invoked repeatedly by the queue manager to retrieve all of the authority data that matches the selection criteria specified on the first invocation.

MQZ_FREE_USER - Free user
This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to free associated allocated resource.

MQZ_GET_AUTHORITY - Get authority
This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that an entity has to access the specified object.

MQZ_GET_AUTHORITY_2 - Get authority (extended)
This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to retrieve the authority that an entity has to access the specified object.

MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority
This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that a named group has to access a specified object (but without the additional authority of the nobody group), or the authority that the primary group of the named principal has to access a specified object.

MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)
This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to retrieve the authority that a named group has to access a specified object (but without the additional authority of the nobody group), or the authority that the primary group of the named principal has to access a specified object.

MQZ_INIT_AUTHORITY - Initialize authorization service
This function is provided by an authorization service component, and is invoked by the queue manager during configuration of the component. It is expected to call MQZEP in order to provide information to the queue manager.

MQZ_INQUIRE - Inquire authorization service
This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to query the supported functionality.

MQZ_REFRESH_CACHE - Refresh all authorizations
This function is provided by an MQZAS_VERSION_3 authorization service component, and is invoked by the queue manager to refresh the list of authorizations held internally by the component.

MQZ_SET_AUTHORITY - Set authority
This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to set the authority that an entity has to access the specified object.

MQZ_SET_AUTHORITY_2 - Set authority (extended)
This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to set the authority that an entity has to access the specified object.

MQZ_TERM_AUTHORITY - Terminate authorization service
This function is provided by an authorization service component, and is invoked by the queue manager when it no longer requires the services of this component. The function must perform any cleanup required by the component.

MQZAC - Application context
The MQZAC structure is used on the MQZ_AUTHENTICATE_USER call for the *ApplicationContext* parameter. This parameter specifies data related to the calling application.

MQZAD - Authority data
The MQZAD structure is used on the MQZ_ENUMERATE_AUTHORITY_DATA call for two parameters, one input and one output.

MQZED - Entity descriptor
The MQZED structure describes the information that is passed to the MQZAS_VERSION_2 authorization service calls.

MQZIC - Identity context
The MQZIC structure is used on the MQZ_AUTHENTICATE_USER call for the *IdentityContext* parameter.

MQZFP - Free parameters
The MQZFP structure is used on the MQZ_FREE_USER call for the *FreeParms*parameter. This parameter specifies data related to resource to be freed.

MQZ_DELETE_NAME - Delete name
This function is provided by a name service component, and is invoked by the queue manager to delete an entry for the specified queue.

MQZ_INIT_NAME - Initialize name service
This function is provided by a name service component, and is invoked by the queue manager during configuration of the component. It is expected to call MQZEP in order to provide information to the queue manager.

MQZ_INSERT_NAME - Insert name
This function is provided by a name service component, and is invoked by the queue manager to insert an entry for the specified queue, containing the name of the queue manager that owns the queue. If the queue is already defined in the service, the call fails.

MQZ_LOOKUP_NAME - Lookup name
This function is provided by a name service component, and is invoked by the queue manager to retrieve the name of the owning queue manager, for a specified queue.

MQZ_TERM_NAME - Terminate name service
This function is provided by a name service component, and is invoked by the queue manager when it no longer requires the services of this component. The function must perform any cleanup required by the component.

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16540_

7.4.1. How the functions are shown

For each function there is a description, including the function identifier (for MQZEP).

The *parameters* are shown listed in the order they must occur. They must all be present.

Parameters and data types

Each parameter name is followed by its data type in parentheses. These are the elementary data types described in the WebSphere MQ Application Programming Reference manual.

The C language invocation is also given, after the description of the parameters.

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:47

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16550_

7.4.2. MQZEP – Add component entry point

This function is invoked by a service component, during initialization, to add an entry point to the entry point vector for that service component.

Syntax
MQZEP *(Hconfig, Function, EntryPoint, CompCode, Reason)*

C invocation

```
MQZEP (Hconfig, Function, EntryPoint, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHCONFIG  Hconfig;      /* Configuration handle */
MQLONG     Function;     /* Function identifier */
PMQFUNC    EntryPoint;   /* Function entry point */
MQLONG     CompCode;     /* Completion code */
MQLONG     Reason;       /* Reason code qualifying CompCode */
```

Parameter descriptions

Hconfig (MQHCONFIG) - input

Function (MQLONG) - input

EntryPoint (PMQFUNC) - input

CompCode (MQLONG) - output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16570_

7.4.2.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the component which is being configured for this particular installable service. It must be the same as the one passed to the component configuration function by the queue manager on the component initialization call.

Parent topic: MQZEP – Add component entry point

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16600_

7.4.2.2. Function (MQLONG) – input

Function identifier.

Valid values for this are defined for each installable service.

If MQZEP is called more than once for the same function, the last call made provides the entry point which is used.

Parent topic: MQZEP – Add component entry point

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16610_

7.4.2.3. EntryPoint (PMQFUNC) – input

Function entry point.

This is the address of the entry point provided by the component to perform the function.

The value NULL is valid, and indicates that the function is not provided by this component. NULL is assumed for entry points which are not defined using MQZEP.

Parent topic: MQZEP – Add component entry point

This build: January 26, 2011 11:55:47

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16620_

7.4.2.4. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZEP – Add component entry point

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16630_

7.4.2.5. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_FUNCTION_ERROR

(2281, X'8E9') Function identifier not valid.

MQRC_HCONFIG_ERROR

(2280, X'8E8') Configuration handle not valid.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZEP – Add component entry point

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16640_

7.4.3. MQZ_AUTHENTICATE_USER – Authenticate user

This function is provided by an MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to authenticate a user, or to set identity context fields. It is invoked when WebSphere® MQ's user application context is established.

The application context is established during connect calls at the point where the application's user context is initialized, and at each point where the application's user context is changed. Each time a connect call is made, the application's user context information is reacquired in the *IdentityContext* field.

The function identifier for this function (for MQZEP) is MQZID_AUTHENTICATE_USER.

Syntax

MQZ_AUTHENTICATE_USER(*QMgrName, SecurityParms, ApplicationContext, IdentityContext, CorrelationPtr, ComponentData, Continuation, CompCode, Reason*)

C invocation

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                       IdentityContext, &CorrelationPtr, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;             /* Queue manager name */
MQCSP      SecurityParms;        /* Security parameters */
MQZAC      ApplicationContext;  /* Application context */
MQZIC      IdentityContext;      /* Identity context */
MQPTR      CorrelationPtr;       /* Correlation pointer */
MQBYTE     ComponentData[n];     /* Component data */
```

```
MQLONG    Continuation;        /* Continuation indicator set by
                                  component */
MQLONG    CompCode;            /* Completion code */
MQLONG    Reason;              /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

SecurityParms (MQCSP) – input

ApplicationContext (MQZAC) – input

IdentityContext (MQZIC) – input/output

CorrelationPtr (MQPTR) – output

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16700_

7.4.3.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_AUTHENTICATE_USER – Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16730_

7.4.3.2. SecurityParms (MQCSP) – input

Security parameters.

Data relating to the user ID, password, and authentication type. If the *AuthenticationType* attribute of the MQCSP structure is specified as MQCSP_AUTH_USER_ID_AND_PWD, both the user ID and password are compared against the equivalent fields in the *IdentityContext* (MQZIC) parameter to determine whether they match. For more information, see the WebSphere MQ Application Programming Reference.

During an MQCONN MQI call this parameter contains null, or default values.

Parent topic: MQZ_AUTHENTICATE_USER – Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16740_

7.4.3.3. ApplicationContext (MQZAC) – input

Application context.

Data relating to the calling application. See MQZAC - Application context for details.

During every MQCONN or MQCONNX MQI call, the user context information in the MQZAC structure is reacquired.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16750_

7.4.3.4. IdentityContext (MQZIC) – input/output

Identity context.

On input to the authenticate user function, this identifies the current identity context. The authenticate user function can change this, at which point the queue manager adopts the new identity context. See MQZIC - Identity context for more details on the MQZIC structure.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16760_

7.4.3.5. CorrelationPtr (MQPTR) – output

Correlation pointer.

Specifies the address of any correlation data. This pointer is subsequently passed on to other OAM calls.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16770_

7.4.3.6. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16780_

7.4.3.7. Continuation (MQLONG) – output

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on other components.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16790_

7.4.3.8. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16800_

7.4.3.9. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
  (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
  (2289, X'8F1') Unexpected error occurred accessing service.

Parent topic: MQZ_AUTHENTICATE_USER - Authenticate user

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16810_

7.4.4. MQZ_CHECK_AUTHORITY – Check authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is

invoked by the queue manager to check whether an entity has authority to perform a particular action, or actions, on a specified object.

The function identifier for this function (for MQZEP) is MQZID_CHECK_AUTHORITY.

Syntax

MQZ_CHECK_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_CHECK_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR12   EntityName;         /* Entity name */
MQLONG     EntityType;         /* Entity type */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQLONG     Authority;          /* Authority to be checked */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityName (MQCHAR12) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:48

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16830_

7.4.4.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16860_

7.4.4.2. EntityName (MQCHAR12) – input

Entity name.

The name of the entity whose authorization to the object is to be checked. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

It is not essential for this entity to be known to the underlying security service. If it is not known, the authorizations of the special nobody group (to which all entities are assumed to belong) are used for the check. An all-blank name is valid and can be used in this way.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:48

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16870_

7.4.4.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityName*. It is one of the following:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16880_

7.4.4.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16890_

7.4.4.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16900_

7.4.4.6. Authority (MQLONG) – input

Authority to be checked.

If one authorization is being checked, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being checked, it is the bitwise OR of the corresponding MQZAO_* constants.

The following authorizations apply to use of the MQI calls:

MQZAO_CONNECT

Ability to use the MQCONN call.

MQZAO_BROWSE

Ability to use the MQGET call with a browse option.

This allows the MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR, or MQGMO_BROWSE_NEXT option to be specified on the MQGET call.

MQZAO_INPUT

Ability to use the MQGET call with an input option.

This allows the MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE, or MQOO_INPUT_AS_Q_DEF option to be specified on the MQOPEN call.

MQZAO_OUTPUT

Ability to use the MQPUT call.

This allows the MQOO_OUTPUT option to be specified on the MQOPEN call.

MQZAO_INQUIRE

Ability to use the MQINQ call.

This allows the MQOO_INQUIRE option to be specified on the MQOPEN call.

MQZAO_SET

Ability to use the MQSET call.

This allows the MQOO_SET option to be specified on the MQOPEN call.

MQZAO_PASS_IDENTITY_CONTEXT

Ability to pass identity context.

This allows the MQOO_PASS_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_PASS_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_PASS_ALL_CONTEXT

Ability to pass all context.

This allows the MQOO_PASS_ALL_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_PASS_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_IDENTITY_CONTEXT

Ability to set identity context.

This allows the MQOO_SET_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_ALL_CONTEXT

Ability to set all context.

This allows the MQOO_SET_ALL_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_ALTERNATE_USER_AUTHORITY

Ability to use alternate user authority.

This allows the MQOO_ALTERNATE_USER_AUTHORITY option to be specified on the MQOPEN call, and the MQPMO_ALTERNATE_USER_AUTHORITY option to be specified on the MQPUT1 call.

MQZAO_ALL_MQI

All of the MQI authorizations.

This enables all of the authorizations described above.

The following authorizations apply to administration of a queue manager:

MQZAO_CREATE

Ability to create objects of a specified type.

MQZAO_DELETE

Ability to delete a specified object.

MQZAO_DISPLAY

Ability to display the attributes of a specified object.

MQZAO_CHANGE

Ability to change the attributes of a specified object.

MQZAO_CLEAR

Ability to delete all messages from a specified queue.

MQZAO_AUTHORIZE

Ability to authorize other users for a specified object.

MQZAO_CONTROL

Ability to start or stop a listener, service, or non-client channel object, and the ability to ping a non-client channel object.

MQZAO_CONTROL_EXTENDED

Ability to reset a sequence number, or resolve an indoubt message on a non-client channel object.

MQZAO_ALL_ADMIN

All of the administration authorizations, other than MQZAO_CREATE.

The following authorizations apply to both use of the MQI and to administration of a queue manager:

MQZAO_ALL

All authorizations, other than MQZAO_CREATE.

MQZAO_NONE

No authorizations.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16910_

7.4.4.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16920_

7.4.4.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_CHECK_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

If the call to a component fails (that is, *CompCode* returns MQCC_FAILED), and the *Continuation* parameter is MQZCI_DEFAULT or MQZCI_CONTINUE, the queue manager continues to call other components if there are any.

If the call succeeds (that is, *CompCode* returns MQCC_OK) no other components are called no matter what the setting of *Continuation* is.

If the call fails and the *Continuation* parameter is MQZCI_STOP then no other components are called and the error is returned to the queue manager. Components have no knowledge of previous calls, so the *Continuation* parameter is always set to MQZCI_DEFAULT prior to the call.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16930_

7.4.4.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16940_

7.4.4.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_CHECK_AUTHORITY - Check authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16950_

7.4.5. MQZ_CHECK_AUTHORITY_2 – Check authority (extended)

This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to check whether an entity has authority to perform a particular action, or actions, on a specified object.

The function identifier for this function (for MQZEP) is MQZID_CHECK_AUTHORITY.

MQZ_CHECK_AUTHORITY_2 is similar to MQZ_CHECK_AUTHORITY, but with the *EntityName* parameter replaced by the *EntityData* parameter.

Syntax

MQZ_CHECK_AUTHORITY_2 *(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_CHECK_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                       ObjectName, ObjectType, Authority, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED     EntityData;         /* Entity data */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority to be checked */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityData (MQZED) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa16970_

7.4.5.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_CHECK_AUTHORITY_2 – Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17000_

7.4.5.2. EntityData (MQZED) – input

Entity data.

Data relating to the entity whose authorization to the object is to be checked. See MQZED – Entity descriptor for details.

It is not essential for this entity to be known to the underlying security service. If it is not known, the authorizations of the special nobody group (to which all entities are assumed to belong) are used for the check. An all-blank name is valid and can be used in this way.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17010_

7.4.5.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityData*. It is one of the following:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17020_

7.4.5.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17030_

7.4.5.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

MQOT_TOPIC

Topics.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17040_

7.4.5.6. Authority (MQLONG) – input

Authority to be checked.

If one authorization is being checked, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being checked, it is the bitwise OR of the corresponding MQZAO_* constants.

The following authorizations apply to use of the MQI calls:

MQZAO_CONNECT

Ability to use the MQCONN call.

MQZAO_BROWSE

Ability to use the MQGET call with a browse option.

This allows the MQGMO_BROWSE_FIRST, MQGMO_BROWSE_MSG_UNDER_CURSOR, or MQGMO_BROWSE_NEXT option to be specified on the MQGET call.

MQZAO_INPUT

Ability to use the MQGET call with an input option.

This allows the MQOO_INPUT_SHARED, MQOO_INPUT_EXCLUSIVE, or MQOO_INPUT_AS_Q_DEF option to be specified on the MQOPEN call.

MQZAO_OUTPUT

Ability to use the MQPUT call.

This allows the MQOO_OUTPUT option to be specified on the MQOPEN call.

MQZAO_INQUIRE

Ability to use the MQINQ call.

This allows the MQOO_INQUIRE option to be specified on the MQOPEN call.

MQZAO_SET

Ability to use the MQSET call.

This allows the MQOO_SET option to be specified on the MQOPEN call.

MQZAO_PASS_IDENTITY_CONTEXT

Ability to pass identity context.

This allows the MQOO_PASS_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_PASS_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_PASS_ALL_CONTEXT

Ability to pass all context.

This allows the MQOO_PASS_ALL_CONTEXT option to be specified on the MQOPEN call, and the

MQPMO_PASS_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_IDENTITY_CONTEXT

Ability to set identity context.

This allows the MQOO_SET_IDENTITY_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_IDENTITY_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_SET_ALL_CONTEXT

Ability to set all context.

This allows the MQOO_SET_ALL_CONTEXT option to be specified on the MQOPEN call, and the MQPMO_SET_ALL_CONTEXT option to be specified on the MQPUT and MQPUT1 calls.

MQZAO_ALTERNATE_USER_AUTHORITY

Ability to use alternate user authority.

This allows the MQOO_ALTERNATE_USER_AUTHORITY option to be specified on the MQOPEN call, and the MQPMO_ALTERNATE_USER_AUTHORITY option to be specified on the MQPUT1 call.

MQZAO_ALL_MQI

All of the MQI authorizations.

This enables all of the authorizations described above.

The following authorizations apply to administration of a queue manager:

MQZAO_CREATE

Ability to create objects of a specified type.

MQZAO_DELETE

Ability to delete a specified object.

MQZAO_DISPLAY

Ability to display the attributes of a specified object.

MQZAO_CHANGE

Ability to change the attributes of a specified object.

MQZAO_CLEAR

Ability to delete all messages from a specified queue.

MQZAO_AUTHORIZE

Ability to authorize other users for a specified object.

MQZAO_CONTROL

Ability to start or stop a listener, service, or non-client channel object, and the ability to ping a non-client channel object.

MQZAO_CONTROL_EXTENDED

Ability to reset a sequence number, or resolve an indoubt message on a non-client channel object.

MQZAO_ALL_ADMIN

All of the administration authorizations, other than MQZAO_CREATE.

The following authorizations apply to both use of the MQI and to administration of a queue manager:

MQZAO_ALL

All authorizations, other than MQZAO_CREATE.

MQZAO_NONE

No authorizations.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17050_

7.4.5.7. ComponentData (MQBYTE×ComponentDataLength) –

input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17060_

7.4.5.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_CHECK_AUTHORITY_2 this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17070_

7.4.5.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17080_

7.4.5.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_CHECK_AUTHORITY_2 - Check authority (extended)

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17090_

7.4.6. MQZ_COPY_ALL_AUTHORITY – Copy all authority

This function is provided by an authorization service component. It is invoked by the queue manager to copy all of the authorizations that are currently in force for a reference object to another object.

The function identifier for this function (for MQZEP) is MQZID_COPY_ALL_AUTHORITY.

Syntax

MQZ_COPY_ALL_AUTHORITY*(QMgrName, RefObjectName, ObjectName, ObjectType, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_COPY_ALL_AUTHORITY (QMgrName, RefObjectName, ObjectName, ObjectType,
                        ComponentData, &Continuation, &CompCode,
                        &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR48   RefObjectName;     /* Reference object name */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQBYTE     ComponentData[n];  /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                    component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

RefObjectName (MQCHAR48) - input

ObjectName (MQCHAR48) - input

ObjectType (MQLONG) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

Continuation (MQLONG) - output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17110_

7.4.6.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:49

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17140_

7.4.6.2. RefObjectName (MQCHAR48) – input

Reference object name.

The name of the reference object, the authorizations for which are to be copied. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17150_

7.4.6.3. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which accesses are to be set. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17160_

7.4.6.4. ObjectType (MQLONG) – input

Object type.

The type of object specified by *RefObjectName* and *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

MQOT_TOPIC

Topic.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17170_

7.4.6.5. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17180_

7.4.6.6. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_COPY_ALL_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17190_

7.4.6.7. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17200_

7.4.6.8. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_REF_OBJECT

(2294, X'8F6') Reference object unknown.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_COPY_ALL_AUTHORITY - Copy all authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17210_

7.4.7. MQZ_DELETE_AUTHORITY – Delete authority

This function is provided by an authorization service component, and is invoked by the queue manager to delete all of the authorizations associated with the specified object.

The function identifier for this function (for MQZEP) is MQZID_DELETE_AUTHORITY.

Syntax

```
MQZ_DELETE_AUTHORITY(QMgrName, ObjectName, ObjectType,ComponentData, Continuation, CompCode,
Reason)
```

C invocation

```
MQZ_DELETE_AUTHORITY (QMgrName, ObjectName, ObjectType, ComponentData,
                      &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17230_

7.4.7.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_DELETE_AUTHORITY – Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17260_

7.4.7.2. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which accesses are to be deleted. The maximum length of the

string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17270_

7.4.7.3. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
  Authentication information.
MQOT_CHANNEL
  Channel.
MQOT_CLNTCONN_CHANNEL
  Client connection channel.
MQOT_LISTENER
  Listener.
MQOT_NAMELIST
  Namelist.
MQOT_PROCESS
  Process definition.
MQOT_Q
  Queue.
MQOT_Q_MGR
  Queue manager.
MQOT_SERVICE
  Service.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17280_

7.4.7.4. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa17290_

7.4.7.5. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on queue manager.

  For MQZ_DELETE_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE
  Continue with next component.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17300_

7.4.7.6. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17310_

7.4.7.7. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
  (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR
  (2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
  (2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_DELETE_AUTHORITY - Delete authority

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17320_

7.4.8. MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This function is provided by an MQZAS_VERSION_4 authorization service component, and is invoked repeatedly by the queue manager to retrieve all of the authority data that matches the selection criteria specified on the first invocation.

The function identifier for this function (for MQZEP) is MQZID_ENUMERATE_AUTHORITY_DATA.

Syntax

MQZ_ENUMERATE_AUTHORITY_DATA*(QMgrName, StartEnumeration, Filter, AuthorityBufferLength, AuthorityBuffer, AuthorityDataLength, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_ENUMERATE_AUTHORITY_DATA (QMgrName, StartEnumeration, &Filter,
                              AuthorityBufferLength,
                              &AuthorityBuffer,
                              &AuthorityDataLength, ComponentData,
                              &Continuation, &CompCode,
                              &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;                /* Queue manager name */
MQLONG     StartEnumeration;        /* Flag indicating whether call should
                                       start enumeration */
MQZAD      Filter;                   /* Filter */
MQLONG     AuthorityBufferLength;   /* Length of AuthorityBuffer */
MQZAD      AuthorityBuffer;          /* Authority data */
MQLONG     AuthorityDataLength;     /* Length of data returned in
                                       AuthorityBuffer */
MQBYTE     ComponentData[n];        /* Component data */
MQLONG     Continuation;             /* Continuation indicator set by
                                       component */
MQLONG     CompCode;                 /* Completion code */
MQLONG     Reason;                   /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

StartEnumeration (MQLONG) – input

Filter (MQZAD) – input

AuthorityBufferLength (MQLONG) – input

AuthorityBuffer (MQZAD) – output

AuthorityDataLength (MQLONG) – output

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17340_

7.4.8.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17370_

7.4.8.2. StartEnumeration (MQLONG) – input

Flag indicating whether call should start enumeration.

This indicates whether the call should start the enumeration of authority data, or continue the enumeration of authority data started by a previous call to MQZ_ENUMERATE_AUTHORITY_DATA. The value is one of the following:

MQZSE_START

Start enumeration.

The call is invoked with this value to start the enumeration of authority data. The *Filter* parameter specifies the selection criteria to be used to select the authority data returned by this and successive calls.

MQZSE_CONTINUE

Continue enumeration.

The call is invoked with this value to continue the enumeration of authority data. The *Filter* parameter is ignored in this case, and can be specified as the null pointer (the selection criteria are determined by the *Filter* parameter specified by the call that had *StartEnumeration* set to MQZSE_START).

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17380_

7.4.8.3. Filter (MQZAD) – input

Filter.

If *StartEnumeration* is MQZSE_START, *Filter* specifies the selection criteria to be used to select the authority data to return. If *Filter* is the null pointer, no selection criteria are used, that is, all authority data is returned. See MQZAD - Authority data for details of the selection criteria that can be used.

If *StartEnumeration* is MQZSE_CONTINUE, *Filter* is ignored, and can be specified as the null pointer.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17390_

7.4.8.4. AuthorityBufferLength (MQLONG) – input

Length of *AuthorityBuffer*.

This is the length in bytes of the *AuthorityBuffer* parameter. The authority buffer must be big enough to accommodate the data to be returned.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17400_

7.4.8.5. AuthorityBuffer (MQZAD) – output

Authority data.

This is the buffer in which the authority data is returned. The buffer must be big enough to accommodate an MQZAD structure, an MQZED structure, plus the longest entity name and longest domain name defined.

Note: This parameter is defined as an MQZAD, as the MQZAD always occurs at the start of the buffer. However, if the buffer is actually declared as an MQZAD, the buffer will be too small – it needs to be bigger than an MQZAD so that it can accommodate the MQZAD, MQZED, plus entity and domain names.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17410_

7.4.8.6. AuthorityDataLength (MQLONG) – output

Length of data returned in *AuthorityBuffer*.

This is the length of the data returned in *AuthorityBuffer*. If the authority buffer is too small, *AuthorityDataLength* is set to the length of the buffer required, and the call returns completion code MQCC_FAILED and reason code MQRC_BUFFER_LENGTH_ERROR.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17420_

7.4.8.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA – Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17430_

7.4.8.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_ENUMERATE_AUTHORITY_DATA this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17440_

7.4.8.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17450_

7.4.8.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_BUFFER_LENGTH_ERROR

(2005, X'7D5') Buffer length parameter not valid.

MQRC_NO_DATA_AVAILABLE

(2379, X'94B') No data available.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_ENUMERATE_AUTHORITY_DATA - Enumerate authority data

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17460_

7.4.9. MQZ_FREE_USER – Free user

This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to free associated allocated resource.

It is invoked when an application has finished running under all user contexts, for example during an MQDISC MQI call.

The function identifier for this function (for MQZEP) is MQZID_FREE_USER.

Syntax

MQZ_FREE_USER*(QMgrName, FreeParms, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_AUTHENTICATE_USER (QMgrName, SecurityParms, ApplicationContext,
                       IdentityContext, CorrelationPtr, ComponentData,
                       &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;             /* Queue manager name */
MQZFP      FreeParms;            /* Resource to be freed */
MQBYTE     ComponentData[n];    /* Component data */
MQLONG     Continuation;         /* Continuation indicator set by
                                       component */
MQLONG     CompCode;             /* Completion code */
MQLONG     Reason;               /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

FreeParms (MQZFP) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

Continuation (MQLONG) - output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17480_

7.4.9.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the

full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_FREE_USER - Free user

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17510_

7.4.9.2. FreeParms (MQZFP) – input

Free parameters.

A structure containing data relating to the resource to be freed. See MQZFP - Free parameters for details.

Parent topic: MQZ_FREE_USER - Free user

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17520_

7.4.9.3. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

Parent topic: MQZ_FREE_USER - Free user

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17530_

7.4.9.4. Continuation (MQLONG) – output

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
: Continuation dependent on other components.

MQZCI_STOP
: Do not continue with next component.

Parent topic: MQZ_FREE_USER - Free user

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17540_

7.4.9.5. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_FREE_USER – Free user

This build: January 26, 2011 11:55:50

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17550_

7.4.9.6. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

Parent topic: MQZ_FREE_USER – Free user

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17560_

7.4.10. MQZ_GET_AUTHORITY – Get authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_AUTHORITY.

Syntax

MQZ_GET_AUTHORITY*(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_GET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, &Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR12   EntityName;         /* Entity name */
MQLONG     EntityType;         /* Entity type */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQLONG     Authority;          /* Authority of entity */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

- QMgrName (MQCHAR48) - input
- EntityName (MQCHAR12) - input
- EntityType (MQLONG) - input
- ObjectName (MQCHAR48) - input
- ObjectType (MQLONG) - input
- Authority (MQLONG) - output
- ComponentData (MQBYTE×ComponentDataLength) - input/output
- Continuation (MQLONG) - output
- CompCode (MQLONG) - output
- Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17580_

7.4.10.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17610_

7.4.10.2. EntityName (MQCHAR12) – input

Entity name.

The name of the entity whose access to the object is to be retrieved. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17620_

7.4.10.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17630_

7.4.10.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17640_

7.4.10.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

MQOT_TOPIC

Topic.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17650_

7.4.10.6. Authority (MQLONG) – output

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17660_

7.4.10.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17670_

7.4.10.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_GET_AUTHORITY this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17680_

7.4.10.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17690_

7.4.10.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_GET_AUTHORITY - Get authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17700_

7.4.11. MQZ_GET_AUTHORITY_2 – Get authority (extended)

This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to retrieve the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_AUTHORITY.

MQZ_GET_AUTHORITY_2 is similar to MQZ_GET_AUTHORITY, but with the *EntityName* parameter replaced by the *EntityData* parameter.

Syntax

MQZ_GET_AUTHORITY_2 *(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_GET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, &Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48  QMgrName;           /* Queue manager name */
MQZED     EntityData;         /* Entity data */
MQLONG    EntityType;         /* Entity type */
MQCHAR48  ObjectName;         /* Object name */
MQLONG    ObjectType;         /* Object type */
MQLONG    Authority;          /* Authority of entity */
MQBYTE    ComponentData[n];   /* Component data */
MQLONG    Continuation;       /* Continuation indicator set by
                                 component */
MQLONG    CompCode;           /* Completion code */
MQLONG    Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityData (MQZED) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – output

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17720_

7.4.11.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_GET_AUTHORITY_2 – Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17750_

7.4.11.2. EntityData (MQZED) – input

Entity data.

Data relating to the entity whose access to the object is to be retrieved. See MQZED - Entity descriptor for details.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17760_

7.4.11.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityData*. The following value can be specified:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17770_

7.4.11.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17780_

7.4.11.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

MQOT_TOPIC

Topic.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17790_

7.4.11.6. Authority (MQLONG) – output

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17800_

7.4.11.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17810_

7.4.11.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_GET_AUTHORITY_2 this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17820_

7.4.11.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17830_

7.4.11.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_GET_AUTHORITY_2 - Get authority (extended)

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17840_

7.4.12. MQZ_GET_EXPLICIT_AUTHORITY – Get explicit authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to retrieve the authority that a named group has to access a specified object (but without the additional authority of the nobody group), or the authority that the primary group of the named principal has to access a specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_EXPLICIT_AUTHORITY.

Syntax

MQZ_GET_EXPLICIT_AUTHORITY*(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_GET_EXPLICIT_AUTHORITY (QMgrName, EntityName, EntityType,
                            ObjectName, ObjectType, &Authority,
                            ComponentData, &Continuation,
                            &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR12   EntityName;         /* Entity name */
MQLONG     EntityType;         /* Entity type */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQLONG     Authority;          /* Authority of entity */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityName (MQCHAR12) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – output

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:51

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17860_

7.4.12.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17890_

7.4.12.2. EntityName (MQCHAR12) – input

Entity name.

The name of the entity whose access to the object is to be retrieved. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17900_

7.4.12.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17910_

7.4.12.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:51

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17920_

7.4.12.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
  Authentication information.
MQOT_CHANNEL
  Channel.
MQOT_CLNTCONN_CHANNEL
  Client connection channel.
MQOT_LISTENER
  Listener.
MQOT_NAMELIST
  Namelist.
MQOT_PROCESS
  Process definition.
MQOT_Q
  Queue.
MQOT_Q_MGR
  Queue manager.
MQOT_SERVICE
  Service.
MQOT_TOPIC
  Topic.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17930_

7.4.12.6. Authority (MQLONG) – output

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17940_

7.4.12.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17950_

7.4.12.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on queue manager.

  For MQZ_GET_EXPLICIT_AUTHORITY this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE
  Continue with next component.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17960_

7.4.12.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17970_

7.4.12.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
  (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED
  (2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR
  (2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
  (2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY - Get explicit authority

This build: January 26, 2011 11:55:52

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa17980_

7.4.13. MQZ_GET_EXPLICIT_AUTHORITY_2 – Get explicit authority (extended)

This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to retrieve the authority that a named group has to access a specified object (but without the additional authority of the nobody group), or the authority that the primary group of the named principal has to access a specified object.

The function identifier for this function (for MQZEP) is MQZID_GET_EXPLICIT_AUTHORITY.

MQZ_GET_EXPLICIT_AUTHORITY_2 is similar to MQZ_GET_EXPLICIT_AUTHORITY, but with the *EntityName* parameter replaced by the *EntityData* parameter.

Syntax

MQZ_GET_EXPLICIT_AUTHORITY_2 *(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_GET_EXPLICIT_AUTHORITY_2 (QMgrName, &EntityData, EntityType,
                              ObjectName, ObjectType, &Authority,
                              ComponentData, &Continuation,
                              &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQZED      EntityData;         /* Entity data */
MQLONG     EntityType;         /* Entity type */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQLONG     Authority;          /* Authority of entity */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

EntityData (MQZED) - input

EntityType (MQLONG) - input

ObjectName (MQCHAR48) - input

ObjectType (MQLONG) - input

Authority (MQLONG) - output

ComponentData (MQBYTE×ComponentDataLength) - input/output

Continuation (MQLONG) - output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18000_

7.4.13.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18030_

7.4.13.2. EntityData (MQZED) – input

Entity data.

Data relating to the entity whose access to the object is to be retrieved. See MQZED - Entity descriptor for details.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18040_

7.4.13.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityData*. The following value can be specified:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18050_

7.4.13.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object for which the entity's authority is to be retrieved. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18060_

7.4.13.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
  Authentication information.
MQOT_CHANNEL
  Channel.
MQOT_CLNTCONN_CHANNEL
  Client connection channel.
MQOT_LISTENER
  Listener.
MQOT_NAMELIST
  Namelist.
MQOT_PROCESS
  Process definition.
MQOT_Q
  Queue.
MQOT_Q_MGR
  Queue manager.
MQOT_SERVICE
  Service.
MQOT_TOPIC
  Topic.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18070_

7.4.13.6. Authority (MQLONG) – output

Authority of entity.

If the entity has one authority, this field is equal to the appropriate authorization operation (MQZAO_* constant). If it has more than one authority, this field is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18080_

7.4.13.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18090_

7.4.13.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_GET_EXPLICIT_AUTHORITY_2 this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18100_

7.4.13.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 - Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa18110_

7.4.13.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_GET_EXPLICIT_AUTHORITY_2 – Get explicit authority (extended)

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18120_

7.4.14. MQZ_INIT_AUTHORITY – Initialize authorization service

This function is provided by an authorization service component, and is invoked by the queue manager during configuration of the component. It is expected to call MQZEP in order to provide information to the queue manager.

The function identifier for this function (for MQZEP) is MQZID_INIT_AUTHORITY.

Syntax

MQZ_INIT_AUTHORITY*(Hconfig, Options, QMgrName, ComponentDataLength, ComponentData, Version, CompCode, Reason)*

C invocation

```
MQZ_INIT_AUTHORITY (Hconfig, Options, QMgrName, ComponentDataLength,
                    ComponentData, &Version, &CompCode,
                    &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;               /* Configuration handle */
MQLONG     Options;               /* Initialization options */
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     ComponentDataLength;   /* Length of component data */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Version;               /* Version number */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

Parameter descriptions

Hconfig (MQHCONFIG) – input

Options (MQLONG) – input

QMgrName (MQCHAR48) - input

ComponentDataLength (MQLONG) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

Version (MQLONG) - input/output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:52

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18140_

7.4.14.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the particular component being initialized. It is to be used by the component when calling the queue manager with the MQZEP function.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18170_

7.4.14.2. Options (MQLONG) – input

Initialization options.

It is one of the following:

MQZIO_PRIMARY
  Primary initialization.

MQZIO_SECONDARY
  Secondary initialization.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18180_

7.4.14.3. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18190_

7.4.14.4. ComponentDataLength (MQLONG) – input

Length of component data.

Length in bytes of the *ComponentData* area. This length is defined in the component configuration data.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18200_

7.4.14.5. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This is initialized to all zeroes before calling the component's primary initialization function. This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions (including the initialization function) provided by this component are preserved, and presented the next time one of this component's functions is called.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18210_

7.4.14.6. Version (MQLONG) – input/output

Version number.

On input to the initialization function, this identifies the *highest* version number that the queue manager supports. The initialization function must change this, if necessary, to the version of the interface which *it* supports. If on return the queue manager does not support the version returned by the component, it calls the component's MQZ_TERM_AUTHORITY function and makes no further use of this component.

The following values are supported:

MQZAS_VERSION_1
: Version 1.

MQZAS_VERSION_2
: Version 2.

MQZAS_VERSION_3
: Version 3.

MQZAS_VERSION_4
: Version 4.

MQZAS_VERSION_5
: Version 5.

Parent topic: MQZ_INIT_AUTHORITY - Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18220_

7.4.14.7. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_INIT_AUTHORITY – Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18230_

7.4.14.8. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') Initialization failed for an undefined reason.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_INIT_AUTHORITY – Initialize authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18240_

7.4.15. MQZ_INQUIRE – Inquire authorization service

This function is provided by a MQZAS_VERSION_5 authorization service component, and is invoked by the queue manager to query the supported functionality.

Where multiple service components are used, service components are called in reverse order to the order they were installed in.

The function identifier for this function (for MQZEP) is MQZID_INQUIRE.

Syntax

MQZ_INQUIRE *(QMgrName, SelectorCount, Selectors, IntAttrCount, IntAttrs, CharAttrLength, CharAttrs, SelectorReturned, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_INQUIRE (QMgrName, SelectorCount, Selectors, IntAttrCount,
```

```
                        &IntAttrs, CharAttrLength, &CharAttrs,
                        SelectorReturned, ComponentData, &Continuation,
                        &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
    MQCHAR48    QMgrName;               /* Queue manager name */
    MQLONG      SelectorCount;          /* Selector count */
    MQLONG      Selectors[n];           /* Selectors */
    MQLONG      IntAttrCount;           /* IntAttrs count */
    MQLONG      IntAttrs[n];            /* Integer attributes */
    MQLONG      CharAttrCount;          /* CharAttrs count */
    MQLONG      CharAttrs[n];           /* Chatacter attributes */
    MQLONG      SelectorReturned[n];    /* Selector returned */
    MQBYTE      ComponentData[n];       /* Component data */
    MQLONG      Continuation;           /* Continuation indicator set by
                                           component */
    MQLONG      CompCode;               /* Completion code */
    MQLONG      Reason;                 /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

SelectorCount (MQLONG) – input

Selectors (MQLONG×SelectorCount) – input

IntAttrCount (MQLONG) – input

IntAttrs (MQLONG×IntAttrCount) – output

CharAttrCount (MQLONG) – input

CharAttrs (MQLONG×CharAttrCount) – output

SelectorReturned (MQLONG×SelectorCount) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18260_

7.4.15.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_INQUIRE – Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18290_

7.4.15.2. SelectorCount (MQLONG) – input

Number of selectors.

The number of selectors supplied in the Selectors parameter.

The value must be between zero and 256.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18300_

7.4.15.3. Selectors (MQLONG×SelectorCount) – input

Selectors.

Array of selectors. Each selector identifies a required attribute and must be of one of the following types:

- MQIACF_* (integer)
- MQCACF_* (character)

Selectors can be specified in any order. The number of selectors in the array is indicated by the SelectorCount parameter.

Integer attributes identified by selectors are returned in the IntAttrs parameter in the same order as they appear in Selectors.

Character attributes identified by selectors are returned in the CharAttrs parameter in the same order as they in appear Selectors.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18310_

7.4.15.4. IntAttrCount (MQLONG) – input

Number of integer attributes.

The number of integer attributes supplied in the IntAttrs parameter.

The value must be between zero and 256.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18320_

7.4.15.5. IntAttrs (MQLONG×IntAttrCount) – output

Integer attributes.

Array of integer attributes. The integer attributes are returned in the same order as the corresponding integer selectors in the Selectors array.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18330_

7.4.15.6. CharAttrCount (MQLONG) – input

Length of the character attributes buffer.

The length in bytes of the CharAttrs parameter.

The value must at least sum of the lengths of the requested character attributes. If no character attributes are requested, zero is a valid value.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:52

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18340_

7.4.15.7. CharAttrs (MQLONG×CharAttrCount) – output

Character attributes buffer.

Buffer containing character attributes, concatenated together. The character attributes are returned in the same order as the corresponding character selectors in the Selectors array.

The length of the buffer is given by the CharAttrCount parameter.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18350_

7.4.15.8. SelectorReturned (MQLONG×SelectorCount) – input

Selector returned.

Array of values identifying which attributes have been returned from the set requested for by the selectors in the Selectors parameter. The number of values in this array is indicated by the SelectorCount parameter. Each value in the array relates to the selector from the corresponding position in the Selectors array. Each value is one of the following:

MQZSL_RETURNED

The attribute requested by the corresponding selector in the Selectors parameter has been returned.

MQZSL_NOT_RETURNED

The attribute requested by the corresponding selector in the Selectors parameter has not been returned.

The array is initialized with all values as *MQZSL_NOT_RETURNED*. When an authorization service component returns an attribute, it sets the appropriate value in the array to *MQZSL_RETURNED*. This allows any other authorization service components, to which the inquire call is made, to identify which attributes have already been returned.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa18360_

7.4.15.9. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18370_

7.4.15.10. Continuation (MQLONG) – output

Continuation flag.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on other components.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18380_

7.4.15.11. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_WARNING
  Partial completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_INQUIRE - Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18390_

7.4.15.12. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_WARNING:

MQRC_CHAR_ATTRS_TOO_SHORT

Not enough space for character attributes.

MQRC_INT_COUNT_TOO_SMALL

Not enough space for integer attributes.

If *CompCode* is MQCC_FAILED:

MQRC_SELECTOR_COUNT_ERROR

Number of selectors is not valid.

MQRC_SELECTOR_ERROR

Attribute selector not valid.

MQRC_SELECTOR_LIMIT_EXCEEDED

Too many selectors specified.

MQRC_INT_ATTR_COUNT_ERROR

Number of integer attributes is not valid.

MQRC_INT_ATTRS_ARRAY_ERROR

Integer attributes array not valid.

MQRC_CHAR_ATTR_LENGTH_ERROR

Number of character attributes is not valid.

MQRC_CHAR_ATTRS_ERROR

Character attributes string is not valid.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

Parent topic: MQZ_INQUIRE – Inquire authorization service

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18400_

7.4.16. MQZ_REFRESH_CACHE – Refresh all authorizations

This function is provided by an MQZAS_VERSION_3 authorization service component, and is invoked by the queue manager to refresh the list of authorizations held internally by the component.

The function identifier for this function (for MQZEP) is MQZID_REFRESH_CACHE (8L).

Syntax

MQZ_REFRESH_CACHE*(QMgrName, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_REFRESH_CACHE (QMgrName, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

ComponentData (MQBYTE*ComponentDataLength) - input/output

Continuation (MQLONG) - output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:53

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18420_

7.4.16.1. QMgrName (MQCHAR48) - input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_REFRESH_CACHE - Refresh all authorizations

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23330_

7.4.16.2. ComponentData (MQBYTE*ComponentDataLength) - input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the ComponentDataLength parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_REFRESH_CACHE - Refresh all authorizations

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23340_

7.4.16.3. Continuation (MQLONG) - output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on queue manager.

  For MQZ_REFRESH_CACHE this has the same effect as MQZCI_CONTINUE.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_REFRESH_CACHE - Refresh all authorizations

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23350_

7.4.16.4. CompCode (MQLONG) - output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_REFRESH_CACHE - Refresh all authorizations

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23360_

7.4.16.5. Reason (MQLONG) - output

Reason code qualifying CompCode.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

For more information on this reason code, see MQRC_SERVICE_ERROR.

Parent topic: MQZ_REFRESH_CACHE - Refresh all authorizations

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23370_

7.4.17. MQZ_SET_AUTHORITY – Set authority

This function is provided by a MQZAS_VERSION_1 authorization service component, and is invoked by the queue manager to set the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_SET_AUTHORITY.

Note: This function overrides any existing authorities. To preserve any existing authorities you must set them again with this function.

Syntax

MQZ_SET_AUTHORITY *(QMgrName, EntityName, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_SET_AUTHORITY (QMgrName, EntityName, EntityType, ObjectName,
                   ObjectType, Authority, ComponentData,
                   &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR12   EntityName;         /* Entity name */
MQLONG     EntityType;         /* Entity type */
MQCHAR48   ObjectName;         /* Object name */
MQLONG     ObjectType;         /* Object type */
MQLONG     Authority;          /* Authority to be checked */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityName (MQCHAR12) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18460_

7.4.17.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_SET_AUTHORITY – Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18490_

7.4.17.2. EntityName (MQCHAR12) – input

Entity name.

The name of the entity whose access to the object is to be set. The maximum length of the string is 12 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18500_

7.4.17.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityName*. The following value can be specified:

MQZAET_PRINCIPAL

Principal.

MQZAET_GROUP

Group.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18510_

7.4.17.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18520_

7.4.17.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO

Authentication information.

MQOT_CHANNEL

Channel.

MQOT_CLNTCONN_CHANNEL

Client connection channel.

MQOT_LISTENER

Listener.

MQOT_NAMELIST

Namelist.

MQOT_PROCESS

Process definition.

MQOT_Q

Queue.

MQOT_Q_MGR

Queue manager.

MQOT_SERVICE

Service.

MQOT_TOPIC

Topic.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18530_

7.4.17.6. Authority (MQLONG) – input

Authority to be checked.

If one authorization is being set, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being set, it is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18540_

7.4.17.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18550_

7.4.17.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on queue manager.

  For MQZ_SET_AUTHORITY this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE
  Continue with next component.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18560_

7.4.17.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18570_

7.4.17.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
  (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED
  (2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR
  (2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
  (2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY
  (2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_SET_AUTHORITY - Set authority

This build: January 26, 2011 11:55:53

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18580_

7.4.18. MQZ_SET_AUTHORITY_2 – Set authority (extended)

This function is provided by a MQZAS_VERSION_2 authorization service component, and is invoked by the queue manager to set the authority that an entity has to access the specified object.

The function identifier for this function (for MQZEP) is MQZID_SET_AUTHORITY.

Note: This function overrides any existing authorities. To preserve any existing authorities you must set them again with this function.

MQZ_SET_AUTHORITY_2 is similar to MQZ_SET_AUTHORITY, but with the *EntityName* parameter replaced by the *EntityData* parameter.

Syntax

MQZ_SET_AUTHORITY_2*(QMgrName, EntityData, EntityType, ObjectName, ObjectType, Authority, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_SET_AUTHORITY_2 (QMgrName, &EntityData, EntityType, ObjectName,
                     ObjectType, Authority, ComponentData,
                     &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;            /* Queue manager name */
MQZED      EntityData;          /* Entity data */
MQLONG     EntityType;          /* Entity type */
MQCHAR48   ObjectName;          /* Object name */
MQLONG     ObjectType;          /* Object type */
MQLONG     Authority;           /* Authority to be checked */
MQBYTE     ComponentData[n];    /* Component data */
MQLONG     Continuation;        /* Continuation indicator set by
                                   component */
MQLONG     CompCode;            /* Completion code */
MQLONG     Reason;              /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

EntityData (MQZED) – input

EntityType (MQLONG) – input

ObjectName (MQCHAR48) – input

ObjectType (MQLONG) – input

Authority (MQLONG) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18600_

7.4.18.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18630_

7.4.18.2. EntityData (MQZED) – input

Entity data.

Data relating to the entity whose access to the object is to be set. See MQZED - Entity descriptor for details.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18640_

7.4.18.3. EntityType (MQLONG) – input

Entity type.

The type of entity specified by *EntityData*. The following value can be specified:

MQZAET_PRINCIPAL
  Principal.

MQZAET_GROUP
  Group.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18650_

7.4.18.4. ObjectName (MQCHAR48) – input

Object name.

The name of the object to which access is required. The maximum length of the string is 48 characters; if it is shorter than that it is padded to the right with blanks. The name is not terminated by a null character.

If *ObjectType* is MQOT_Q_MGR, this name is the same as *QMgrName*.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18660_

7.4.18.5. ObjectType (MQLONG) – input

Object type.

The type of entity specified by *ObjectName*. It is one of the following:

MQOT_AUTH_INFO
  Authentication information.
MQOT_CHANNEL
  Channel.
MQOT_CLNTCONN_CHANNEL
  Client connection channel.
MQOT_LISTENER
  Listener.
MQOT_NAMELIST
  Namelist.
MQOT_PROCESS
  Process definition.
MQOT_Q
  Queue.
MQOT_Q_MGR
  Queue manager.
MQOT_SERVICE
  Service.
MQOT_TOPIC
  Topic.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18670_

7.4.18.6. Authority (MQLONG) – input

Authority to be checked.

If one authorization is being set, this field is equal to the appropriate authorization operation (MQZAO_* constant). If more than one authorization is being set, it is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18680_

7.4.18.7. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_AUTHORITY call.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18690_

7.4.18.8. Continuation (MQLONG) – output

Continuation indicator set by component.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

For MQZ_SET_AUTHORITY_2 this has the same effect as MQZCI_STOP.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18700_

7.4.18.9. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:53

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18710_

7.4.18.10. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_NOT_AUTHORIZED

(2035, X'7F3') Not authorized for access.

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_ENTITY

(2292, X'8F4') Entity unknown to service.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_SET_AUTHORITY_2 - Set authority (extended)

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18720_

7.4.19. MQZ_TERM_AUTHORITY – Terminate authorization service

This function is provided by an authorization service component, and is invoked by the queue manager when it no longer requires the services of this component. The function must perform any cleanup required by the component.

The function identifier for this function (for MQZEP) is MQZID_TERM_AUTHORITY.

Syntax

MQZ_TERM_AUTHORITY*(Hconfig, Options, QMgrName, ComponentData, CompCode, Reason)*

C invocation

```
MQZ_TERM_AUTHORITY (Hconfig, Options, QMgrName, ComponentData,
                    &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;             /* Configuration handle */
MQLONG     Options;             /* Termination options */
MQCHAR48   QMgrName;            /* Queue manager name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     CompCode;            /* Completion code */
MQLONG     Reason;              /* Reason code qualifying CompCode */
```

Parameter descriptions

Hconfig (MQHCONFIG) - input

Options (MQLONG) - input

QMgrName (MQCHAR48) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18740_

7.4.19.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the particular component being terminated.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18770_

7.4.19.2. Options (MQLONG) – input

Termination options.

It is one of the following:

MQZTO_PRIMARY

Primary termination.

MQZTO_SECONDARY

Secondary termination.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18780_

7.4.19.3. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the authorization service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18790_

7.4.19.4. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter on the MQZ_INIT_AUTHORITY call.

When the MQZ_TERM_AUTHORITY call has completed, the queue manager discards this data.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18800_

7.4.19.5. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18810_

7.4.19.6. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_TERMINATION_FAILED

(2287, X'8FF') Termination failed for an undefined reason.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_TERM_AUTHORITY - Terminate authorization service

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18820_

7.4.20. MQZAC – Application context

The MQZAC structure is used on the MQZ_AUTHENTICATE_USER call for the *ApplicationContext* parameter. This parameter specifies data related to the calling application.

C declaration

```
typedef struct tagMQZAC MQZAC;
struct tagMQZAC {
  MQCHAR4    StrucId;              /* Structure identifier */
  MQLONG     Version;              /* Structure version number */
  MQPID      ProcessId;            /* Process identifier */
  MQTID      ThreadId;             /* Thread identifier */
  MQCHAR28   ApplName;             /* Application name */
```

```
  MQCHAR12  UserID;              /* User identifier */
  MQCHAR12  EffectiveUserID;    /* Effective user identifier */
  MQLONG    Environment;         /* Environment */
  MQLONG    CallerType;          /* Caller type */
  MQLONG    AuthenticationType; /* Authentication type */
  MQLONG    BindType;            /* Bind type */
};
```

Fields

StrucId (MQCHAR4)

Version (MQLONG)

ProcessId (MQPID)

ThreadId (MQTID)

ApplName (MQCHAR28)

UserID (MQCHAR12)

EffectiveUserID (MQCHAR12)

Environment (MQLONG)

CallerType (MQLONG)

AuthenticationType (MQLONG)

BindType (MQLONG)

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18840_

7.4.20.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQZAC_STRUC_ID

Identifier for application context structure.

For the C programming language, the constant MQZAC_STRUC_ID_ARRAY is also defined; this has the same value as MQZAC_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18860_

7.4.20.2. Version (MQLONG)

Structure version number.

The value is:

MQZAC_VERSION_1

Version-1 application context structure.

The following constant specifies the version number of the current version:

MQZAC_CURRENT_VERSION

Current version of application context structure.

This is an input field to the service.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18870_

7.4.20.3. ProcessId (MQPID)

Process identifier.

The process identifier of the application.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18880_

7.4.20.4. ThreadId (MQTID)

Thread identifier.

The thread identifier of the application.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18890_

7.4.20.5. ApplName (MQCHAR28)

Application name.

The application name.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18900_

7.4.20.6. UserID (MQCHAR12)

User identifier.

On UNIX systems this field specifies the application's real user ID. On Windows this field specifies the application's user ID.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18910_

7.4.20.7. EffectiveUserID (MQCHAR12)

Effective user identifier.

On UNIX systems this field specifies the application's effective user ID. On Windows this field is blank.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18920_

7.4.20.8. Environment (MQLONG)

Environment.

This field specifies the environment from which the call was made.

The value is one of the following:

MQXE_COMMAND_SERVER
  Command server.

MQXE_MQSC
  runmqsc command interpreter.

MQXE_MCA
  Message channel agent

MQXE_OTHER
  Undefined environment

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18930_

7.4.20.9. CallerType (MQLONG)

Caller Type.

This field specifies the type of program that made the call.

The value is one of the following:

MQXACT_EXTERNAL
  The call is external to the queue manager.

MQXACT_INTERNAL
  The call is internal to the queue manager.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa18940_

7.4.20.10. AuthenticationType (MQLONG)

Authentication Type.

This field specifies the type of authentication being performed.

The value is one of the following:

MQZAT_INITIAL_CONTEXT

The authentication call is due to user context being initialized. This value is used during an MQCONN or MQCONNX call.

MQZAT_CHANGE_CONTEXT

The authentication call is due to the user context being changed. This value is used when the MCA changes the user context.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18950_

7.4.20.11. BindType (MQLONG)

Bind Type.

This field specifies the type of binding in use.

The value is one of the following:

MQCNO_FASTPATH_BINDING

Fastpath binding.

MQCNO_SHARED_BINDING

Shared binding.

MQCNO_ISOLATED_BINDING

Isolated binding.

Parent topic: MQZAC - Application context

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18960_

7.4.21. MQZAD – Authority data

The MQZAD structure is used on the MQZ_ENUMERATE_AUTHORITY_DATA call for two parameters, one input and one output.

- MQZAD is used for the *Filter* parameter which is input to the call. This parameter specifies the selection criteria that are to be used to select the authority data returned by the call.
- MQZAD is also used for the *AuthorityBuffer* parameter which is output from the call. This parameter specifies the authorizations for one combination of profile name, object type, and entity.

C declaration

```
typedef struct tagMQZAD MQZAD;
struct tagMQZAD {
  MQCHAR4    StrucId;         /* Structure identifier */
  MQLONG     Version;         /* Structure version number */
  MQCHAR48   ProfileName;     /* Profile name */
```

```
  MQLONG     ObjectType;      /* Object type */
  MQLONG     Authority;       /* Authority */
  PMQZED     EntityDataPtr;   /* Address of MQZED structure identifying an
                                 entity */
  MQLONG     EntityType;      /* Entity type */
  MQAUTHOPT  Options;         /* Options */
};
```

Fields

StrucId (MQCHAR4)

Version (MQLONG)

ProfileName (MQCHAR48)

ObjectType (MQLONG)

Authority (MQLONG)

EntityDataPtr (PMQZED)

EntityType (MQLONG)

Options (MQAUTHOPT)

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa18980_

7.4.21.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQZAD_STRUC_ID
  Identifier for authority data structure.

  For the C programming language, the constant MQZAD_STRUC_ID_ARRAY is also defined; this has the same value as MQZAD_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19000_

7.4.21.2. Version (MQLONG)

Structure version number.

The value is:

MQZAD_VERSION_1
  Version-1 authority data structure.

The following constant specifies the version number of the current version:

MQZAD_CURRENT_VERSION

Current version of authority data structure.

This is an input field to the service.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19010_

7.4.21.3. ProfileName (MQCHAR48)

Profile name.

For the *Filter* parameter, this field is the profile name whose authority data is required. If the name is entirely blank up to the end of the field or the first null character, authority data for all profile names is returned.

For the *AuthorityBuffer* parameter, this field is the name of a profile that matches the specified selection criteria.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19020_

7.4.21.4. ObjectType (MQLONG)

Object type.

For the *Filter* parameter, this field is the object type for which authority data is required. If the value is MQOT_ALL, authority data for all object types is returned.

For the *AuthorityBuffer* parameter, this field is the object type to which the profile identified by *ProfileName* applies.

The value is one of the following; for the *Filter* parameter, the value MQOT_ALL is also valid:

MQOT_AUTH_INFO
: Authentication information.

MQOT_CHANNEL
: Channel.

MQOT_CLNTCONN_CHANNEL
: Client connection channel.

MQOT_LISTENER
: Listener.

MQOT_NAMELIST
: Namelist.

MQOT_PROCESS
: Process definition.

MQOT_Q
: Queue.

MQOT_Q_MGR
: Queue manager.

MQOT_SERVICE
: Service.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19030_

7.4.21.5. Authority (MQLONG)

Authority.

For the *Filter* parameter, this field is ignored.

For the *AuthorityBuffer* parameter, this field represents the authorizations that the entity has to the objects identified by *ProfileName* and *ObjectType*. If the entity has only one authority, the field is equal to the appropriate authorization value (MQZAO_* constant). If the entity has more than one authority, the field is the bitwise OR of the corresponding MQZAO_* constants.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19040_

7.4.21.6. EntityDataPtr (PMQZED)

Address of MQZED structure identifying an entity.

For the *Filter* parameter, this field points to an MQZED structure that identifies the entity whose authority data is required. If *EntityDataPtr* is the null pointer, authority data for all entities is returned.

For the *AuthorityBuffer* parameter, this field points to an MQZED structure that identifies the entity whose authority data has been returned.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19050_

7.4.21.7. EntityType (MQLONG)

Entity type.

For the *Filter* parameter, this field specifies the entity type for which authority data is required. If the value is MQZAET_NONE, authority data for all entity types is returned.

For the *AuthorityBuffer* parameter, this field specifies the type of the entity identified by the MQZED structure pointed to by *EntityDataPtr*.

The value is one of the following; for the *Filter* parameter, the value MQZAET_NONE is also valid:

MQZAET_PRINCIPAL
: Principal.

MQZAET_GROUP
: Group.

Parent topic: MQZAD - Authority data

This build: January 26, 2011 11:55:54

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa19060_

7.4.21.8. Options (MQAUTHOPT)

Options.

This field specifies options that give control over the profiles that are displayed.

One of the following must be specified:

MQAUTHOPT_NAME_ALL_MATCHING

Displays all profiles

MQAUTHOPT_NAME_EXPLICIT

Displays profiles that have exactly the same name as specified in the *ProfileName* field.

In addition, one of the following must also be specified:

MQAUTHOPT_ENTITY_SET

Display all profiles used to calculate the cumulative authority that the entity has to the object specified by *ProfileName*. The *ProfileName* field must not contain any wildcard characters.

- If the specified entity is a principal, for each member of the set {entity, groups} the most applicable profile that applies to the object is displayed.
- If the specified entity is a group, the most applicable profile from the group that applies to the object is displayed.
- If this value is specified, then the values of *ProfileName*, *ObjectType*, *EntityType*, and the entity name specified in the *EntityDataPtr* MQZED structure, must all be non-blank.

If you have specified *MQAUTHOPT_NAME_ALL_MATCHING*, you can also specify the following:

MQAUTHOPT_ENTITY_EXPLICIT

Displays profiles that have exactly the same entity name as the entity name specified in the *EntityDataPtr* MQZED structure.

Parent topic: MQZAD – Authority data

This build: January 26, 2011 11:55:54

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19070_

7.4.22. MQZED – Entity descriptor

The MQZED structure describes the information that is passed to the MQZAS_VERSION_2 authorization service calls.

C declaration

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  PMQCHAR    EntityNamePtr;     /* Address of entity name */
  PMQCHAR    EntityDomainPtr;   /* Address of entity domain name */
  MQBYTE40   SecurityId;        /* Security identifier */
  MQPTR      CorrelationPtr;    /* Address of correlation data */
```

Fields

StrucId (MQCHAR4)

Version (MQLONG)

EntityNamePtr (PMQCHAR)

EntityDomainPtr (PMQCHAR)

SecurityId (MQBYTE40)

CorrelationPtr (MQPTR)

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19090_

7.4.22.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQZED_STRUC_ID

Identifier for entity descriptor structure.

For the C programming language, the constant MQZED_STRUC_ID_ARRAY is also defined; this has the same value as MQZED_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Parent topic: MQZED - Entity descriptor

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19110_

7.4.22.2. Version (MQLONG)

Structure version number.

The value is:

MQZED_VERSION_1

Version-1 entity descriptor structure.

The following constant specifies the version number of the current version:

MQZED_CURRENT_VERSION

Current version of entity descriptor structure.

This is an input field to the service.

Parent topic: MQZED - Entity descriptor

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19120_

7.4.22.3. EntityNamePtr (PMQCHAR)

Address of entity name.

This is a pointer to the name of the entity whose authorization is to be checked.

Parent topic: MQZED - Entity descriptor

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19130_

7.4.22.4. EntityDomainPtr (PMQCHAR)

Address of entity domain name.

This is a pointer to the name of the domain containing the definition of the entity whose authorization is to be checked.

Parent topic: MQZED – Entity descriptor

This build: January 26, 2011 11:55:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19140_

7.4.22.5. SecurityId (MQBYTE40)

Security identifier.

This is the security identifier whose authorization is to be checked.

Parent topic: MQZED – Entity descriptor

This build: January 26, 2011 11:55:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19150_

7.4.22.6. CorrelationPtr (MQPTR)

Correlation pointer.

This facilitates the passing of correlational data between the authenticate user function and other appropriate OAM functions.

Parent topic: MQZED – Entity descriptor

This build: January 26, 2011 11:55:55

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19160_

7.4.23. MQZIC – Identity context

The MQZIC structure is used on the MQZ_AUTHENTICATE_USER call for the *IdentityContext* parameter.

The MQZIC structure contains identity context information, that identifies the user of the application that first put the message on a queue:

- The queue manager fills the UserIdentifier field with a name that identifies the user, the way that the queue manager can do this depends on the environment in which the application is running.
- The queue manager fills the AccountingToken field with a token or number that it determined from the application that put the message.
- Applications can use the ApplIdentityData field for any extra information that they want to include about the user (for example, an encrypted password).

Suitably authorized applications may set the identity context using the MQZ_AUTHENTICTAE_USER function.

A Windows systems security identifier (SID) is stored in the AccountingToken field when a message is created under WebSphere® MQ for Windows. The SID can be used to supplement the UserIdentifier field and to establish the credentials of a user.

C declaration

```
typedef struct tagMQZED MQZED;
struct tagMQZED {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQCHAR12   UserIdentifier;    /* User identifier */
  MQBYTE32   AccountingToken;   /* Accounting token */
  MQCHAR32   ApplIdentityData; /* Application data relating to identity */
};
```

Fields

StrucId (MQCHAR4)

Version (MQLONG)

UserIdentifier (MQCHAR12)

AccountingToken (MQBYTE32)

ApplIdentityData (MQCHAR32)

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19180_

7.4.23.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQZIC_STRUC_ID

Identifier for identity context structure.

For the C programming language, the constant MQZIC_STRUC_ID_ARRAY is also defined; this has the same value as MQZIC_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Parent topic: MQZIC - Identity context

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19200_

7.4.23.2. Version (MQLONG)

Structure version number.

The value is:

MQZIC_VERSION_1

Version-1 identity context structure.

The following constant specifies the version number of the current version:

MQZIC_CURRENT_VERSION

Current version of identity context structure.

This is an input field to the service.

Parent topic: MQZIC - Identity context

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19210_

7.4.23.3. UserIdentifier (MQCHAR12)

User identifier.

This is part of the identity context of the message.

UserIdentifier specifies the user identifier of the application that originated the message. The queue manager treats this information as character data, but does not define the format of it. For more information on the *UserIdentifier* field, see WebSphere MQ Application Programming Reference.

Parent topic: MQZIC - Identity context

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19220_

7.4.23.4. AccountingToken (MQBYTE32)

Accounting token.

This is part of the identity context of the message.

AccountingToken allows an application to cause work done as a result of the message to be appropriately charged. The queue manager treats this information as a string of bits and does not check its content. For more information on the *AccountingToken* field, see WebSphere MQ Application Programming Reference.

Parent topic: MQZIC - Identity context

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19230_

7.4.23.5. ApplIdentityData (MQCHAR32)

Application data relating to identity.

This is part of the identity context of the message.

ApplIdentityData is information that is defined by the application suite that can be used to provide additional information about the origin of the message. For example, it could be set by applications running with suitable user authority to indicate whether the identity data is trusted. For more information on the *ApplIdentityData* field, see WebSphere MQ Application Programming Reference.

Parent topic: MQZIC - Identity context

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19240_

7.4.24. MQZFP – Free parameters

The MQZFP structure is used on the MQZ_FREE_USER call for the *FreeParms*parameter. This parameter specifies data related to resource to be freed.

C declaration

```
typedef struct tagMQZFP MQZFP;
struct tagMQZFP {
  MQCHAR4    StrucId;           /* Structure identifier */
  MQLONG     Version;           /* Structure version number */
  MQBYTE8    Reserved;          /* Reserved field */
  MQPTR      CorrelationPtr;    /* Address of correlation data */
};
```

Fields

StrucId (MQCHAR4)

Version (MQLONG)

Reserved (MQBYTE8)

CorrelationPtr (MQPTR)

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19260_

7.4.24.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQZFP_STRUC_ID
  Identifier for free parameters structure.

  For the C programming language, the constant MQZFP_STRUC_ID_ARRAY is also defined; this has the same value as MQZFP_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the service.

Parent topic: MQZFP – Free parameters

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19280_

7.4.24.2. Version (MQLONG)

Structure version number.

The value is:

MQZFP_VERSION_1
  Version-1 free parameters structure.

The following constant specifies the version number of the current version:

MQZFP_CURRENT_VERSION
  Current version of free parameters structure.

This is an input field to the service.

Parent topic: MQZFP - Free parameters

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19290_

7.4.24.3. Reserved (MQBYTE8)

Reserved field.

The initial value is null.

Parent topic: MQZFP - Free parameters

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19300_

7.4.24.4. CorrelationPtr (MQPTR)

Correlation pointer.

Address of correlation data relating to the resource to be freed.

Parent topic: MQZFP - Free parameters

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19310_

7.4.25. MQZ_DELETE_NAME – Delete name

This function is provided by a name service component, and is invoked by the queue manager to delete an entry for the specified queue.

The function identifier for this function (for MQZEP) is MQZID_DELETE_NAME.

Syntax

MQZ_DELETE_NAME*(QMgrName, QName, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_DELETE_NAME (QMgrName, QName, ComponentData, &Continuation,
                 &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR48   QName;              /* Queue name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

QName (MQCHAR48) - input

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19330_

7.4.25.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the name service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_DELETE_NAME – Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19360_

7.4.25.2. QName (MQCHAR48) – input

Queue name.

The name of the queue for which an entry is to be deleted. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

Parent topic: MQZ_DELETE_NAME – Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19370_

7.4.25.3. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_NAME call.

Parent topic: MQZ_DELETE_NAME – Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa19380_

7.4.25.4. Continuation (MQLONG) – output

Continuation indicator set by component.

For MQZ_DELETE_NAME, the queue manager does not attempt to invoke another component, whatever is returned in *Continuation*.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_DELETE_NAME - Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19390_

7.4.25.5. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_WARNING

Warning (partial completion).

MQCC_FAILED

Call failed.

Parent topic: MQZ_DELETE_NAME - Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19400_

7.4.25.6. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_WARNING:

MQRC_UNKNOWN_Q_NAME

(2288, X'8F0') Queue name not found.

Note: It might not be possible to return this code if the underlying service responds with success for this case.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_DELETE_NAME - Delete name

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19410_

7.4.26. MQZ_INIT_NAME – Initialize name service

This function is provided by a name service component, and is invoked by the queue manager during configuration of the component. It is expected to call MQZEP in order to provide information to the queue manager.

The function identifier for this function (for MQZEP) is MQZID_INIT_NAME.

Syntax

MQZ_INIT_NAME(*Hconfig, Options, QMgrName, ComponentDataLength, ComponentData, Version, CompCode, Reason*)

C invocation

```
MQZ_INIT_NAME (Hconfig, Options, QMgrName, ComponentDataLength,
               ComponentData, &Version, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;               /* Configuration handle */
MQLONG     Options;               /* Initialization options */
MQCHAR48   QMgrName;              /* Queue manager name */
MQLONG     ComponentDataLength;   /* Length of component data */
MQBYTE     ComponentData[n];      /* Component data */
MQLONG     Version;               /* Version number */
MQLONG     CompCode;              /* Completion code */
MQLONG     Reason;                /* Reason code qualifying CompCode */
```

Parameter descriptions

Hconfig (MQHCONFIG) - input

Options (MQLONG) - input

QMgrName (MQCHAR48) - input

ComponentDataLength (MQLONG) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

Version (MQLONG) - input/output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19430_

7.4.26.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the particular component being initialized. It is to be used by the component when calling the queue manager with the MQZEP function.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19460_

7.4.26.2. Options (MQLONG) – input

Initialization options.

It is one of the following:

MQZIO_PRIMARY

Primary initialization.

MQZIO_SECONDARY

Secondary initialization.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19470_

7.4.26.3. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the name service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19480_

7.4.26.4. ComponentDataLength (MQLONG) – input

Length of component data.

Length in bytes of the *ComponentData* area. This length is defined in the component configuration data.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19490_

7.4.26.5. ComponentData (MQBYTE×ComponentDataLength) –

input/output

Component data.

This is initialized to all zeroes before calling the component's primary initialization function. This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions (including the initialization function) provided by this component are preserved, and presented the next time one of this component's functions is called.

Component data is in shared memory accessible to all processes. Therefore primary initialization is the first process initialization and secondary initialization is any subsequent process initialization.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:55

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19500_

7.4.26.6. Version (MQLONG) – input/output

Version number.

On input to the initialization function, this identifies the *highest* version number that the queue manager supports. The initialization function must change this, if necessary, to the version of the interface which *it* supports. If on return the queue manager does not support the version returned by the component, it calls the component's MQZ_TERM_NAME function and makes no further use of this component.

The following value is supported:

MQZNS_VERSION_1

Version 1.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19510_

7.4.26.7. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19520_

7.4.26.8. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_INITIALIZATION_FAILED

(2286, X'8EE') Initialization failed for an undefined reason.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_INIT_NAME - Initialize name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19530_

7.4.27. MQZ_INSERT_NAME – Insert name

This function is provided by a name service component, and is invoked by the queue manager to insert an entry for the specified queue, containing the name of the queue manager that owns the queue. If the queue is already defined in the service, the call fails.

The function identifier for this function (for MQZEP) is MQZID_INSERT_NAME.

Syntax

MQZ_INSERT_NAME*(QMgrName, QName, ResolvedQMgrName, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_INSERT_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR48   QName;              /* Queue name */
MQCHAR48   ResolvedQMgrName;   /* Resolved queue manager name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) - input

QName (MQCHAR48) - input

ResolvedQMgrName (MQCHAR48) - input

ComponentData (MQBYTE×ComponentDataLength) - input/output

Continuation (MQLONG) - output

CompCode (MQLONG) - output

Reason (MQLONG) - output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19550_

7.4.27.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the name service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19580_

7.4.27.2. QName (MQCHAR48) – input

Queue name.

The name of the queue for which an entry is to be inserted. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19590_

7.4.27.3. ResolvedQMgrName (MQCHAR48) – input

Resolved queue manager name.

The name of the queue manager to which the queue resolves. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19600_

7.4.27.4. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_NAME call.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19610_

7.4.27.5. Continuation (MQLONG) – output

Continuation indicator set by component.

For MQZ_INSERT_NAME, the queue manager does not attempt to invoke another component, whatever is returned in *Continuation*.

The following values can be specified:

MQZCI_DEFAULT
  Continuation dependent on queue manager.

MQZCI_STOP
  Do not continue with next component.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19620_

7.4.27.6. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK
  Successful completion.

MQCC_FAILED
  Call failed.

Parent topic: MQZ_INSERT_NAME - Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19630_

7.4.27.7. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE
  (0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_Q_ALREADY_EXISTS
  (2290, X'8F2') Queue object already exists.

MQRC_SERVICE_ERROR
  (2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE
  (2285, X'8ED') Underlying service not available.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_INSERT_NAME – Insert name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19640_

7.4.28. MQZ_LOOKUP_NAME – Lookup name

This function is provided by a name service component, and is invoked by the queue manager to retrieve the name of the owning queue manager, for a specified queue.

The function identifier for this function (for MQZEP) is MQZID_LOOKUP_NAME.

Syntax

MQZ_LOOKUP_NAME *(QMgrName, QName, ResolvedQMgrName, ComponentData, Continuation, CompCode, Reason)*

C invocation

```
MQZ_LOOKUP_NAME (QMgrName, QName, ResolvedQMgrName, ComponentData,
                 &Continuation, &CompCode, &Reason);
```

The parameters passed to the service are declared as follows:

```
MQCHAR48   QMgrName;           /* Queue manager name */
MQCHAR48   QName;              /* Queue name */
MQCHAR48   ResolvedQMgrName;   /* Resolved queue manager name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     Continuation;       /* Continuation indicator set by
                                  component */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

QMgrName (MQCHAR48) – input

QName (MQCHAR48) – input

ResolvedQMgrName (MQCHAR48) – output

ComponentData (MQBYTE×ComponentDataLength) – input/output

Continuation (MQLONG) – output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19660_

7.4.28.1. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the name service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19690_

7.4.28.2. QName (MQCHAR48) – input

Queue name.

The name of the queue which is to be resolved. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19700_

7.4.28.3. ResolvedQMgrName (MQCHAR48) – output

Resolved queue manager name.

If the function completes successfully, this is the name of the queue manager that owns the queue.

The name returned by the service component must be padded on the right with blanks to the full length of the parameter; the name *must not* be terminated by a null character, or contain leading or embedded blanks.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19710_

7.4.28.4. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

Component data is in shared memory accessible to all processes.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter of the MQZ_INIT_NAME call.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19720_

7.4.28.5. Continuation (MQLONG) – output

Continuation indicator set by component.

For MQZ_LOOKUP_NAME, the queue manager specifies whether to invoke another name service

component, as follows:

- If *CompCode* is MQCC_OK, no further components are invoked, whatever value is returned in *Continuation*.
- If *CompCode* is not MQCC_OK, a further component is invoked, unless *Continuation* is MQZCI_STOP. Do not set this value without good reason.

The following values can be specified:

MQZCI_DEFAULT

Continuation dependent on queue manager.

MQZCI_CONTINUE

Continue with next component.

MQZCI_STOP

Do not continue with next component.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19730_

7.4.28.6. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19740_

7.4.28.7. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_ERROR

(2289, X'8F1') Unexpected error occurred accessing service.

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_UNKNOWN_Q_NAME

(2288, X'8F0') Queue name not found.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_LOOKUP_NAME - Lookup name

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19750_

7.4.29. MQZ_TERM_NAME – Terminate name service

This function is provided by a name service component, and is invoked by the queue manager when it no longer requires the services of this component. The function must perform any cleanup required by the component.

The function identifier for this function (for MQZEP) is MQZID_TERM_NAME.

Syntax

MQZ_TERM_NAME*(Hconfig, Options, QMgrName, ComponentData, CompCode, Reason)*

C invocation

```
MQZ_TERM_NAME (Hconfig, Options, QMgrName, ComponentData, &CompCode,
               &Reason);
```

The parameters passed to the service are declared as follows:

```
MQHCONFIG  Hconfig;            /* Configuration handle */
MQLONG     Options;            /* Termination options */
MQCHAR48   QMgrName;           /* Queue manager name */
MQBYTE     ComponentData[n];   /* Component data */
MQLONG     CompCode;           /* Completion code */
MQLONG     Reason;             /* Reason code qualifying CompCode */
```

Parameter descriptions

Hconfig (MQHCONFIG) – input

Options (MQLONG) – input

QMgrName (MQCHAR48) – input

ComponentData (MQBYTE×ComponentDataLength) – input/output

CompCode (MQLONG) – output

Reason (MQLONG) – output

Parent topic: Installable services interface reference information

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19770_

7.4.29.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the particular component being terminated.

Parent topic: MQZ_TERM_NAME – Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19800_

7.4.29.2. Options (MQLONG) – input

Termination options.

It is one of the following:

MQZTO_PRIMARY

Primary termination.

MQZTO_SECONDARY

Secondary termination.

Parent topic: MQZ_TERM_NAME - Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19810_

7.4.29.3. QMgrName (MQCHAR48) – input

Queue manager name.

The name of the queue manager calling the component. This name is padded with blanks to the full length of the parameter; the name is not terminated by a null character.

The queue-manager name is passed to the component for information; the name service interface does not require the component to make use of it in any defined manner.

Parent topic: MQZ_TERM_NAME - Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19820_

7.4.29.4. ComponentData (MQBYTE×ComponentDataLength) – input/output

Component data.

This data is kept by the queue manager on behalf of this particular component; any changes made to it by any of the functions provided by this component are preserved, and presented the next time one of this component's functions is called.

Component data is in shared memory accessible to all processes.

The length of this data area is passed by the queue manager in the *ComponentDataLength* parameter on the MQZ_INIT_NAME call.

When the MQZ_TERM_NAME call has completed, the queue manager discards this data.

Parent topic: MQZ_TERM_NAME - Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19830_

7.4.29.5. CompCode (MQLONG) – output

Completion code.

It is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: MQZ_TERM_NAME - Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19840_

7.4.29.6. Reason (MQLONG) – output

Reason code qualifying *CompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_SERVICE_NOT_AVAILABLE

(2285, X'8ED') Underlying service not available.

MQRC_TERMINATION_FAILED

(2287, X'8FF') Termination failed for an undefined reason.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: MQZ_TERM_NAME - Terminate name service

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19850_

7.5. API exits

API exits let you write code that changes the behavior of WebSphere® MQ API calls, such as MQPUT and MQGET, and then insert that code immediately before or immediately after those calls. The insertion is automatic; the queue manager drives the exit code at the registered points.

This chapter explains why you might want to use API exits, then describes what administration tasks are involved in enabling them. The sections are:

- Why use API exits
- How you use API exits
- What happens when an API exit runs?
- Configuring API exits

We give a brief introduction to writing API exits in How to write an API exit. For detailed information about writing API exits, aimed at application programmers, see the WebSphere MQ Application Programming Guide.

Why use API exits

How you use API exits

What happens when an API exit runs?

Configuring API exits

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:56

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19870_

7.5.1. Why use API exits

There are many reasons why you might want to insert code that modifies the behavior of applications at the level of the queue manager. Each of your applications has a specific job to do, and its code should do that task as efficiently as possible. At a higher level, you might want to apply standards or business processes to a particular queue manager for all the applications that use that queue manager. It is more efficient to do this above the level of individual applications, and thus without having to change the code of each application affected.

Here are a few suggestions of areas in which API exits might be useful:

- For *security*, you can provide authentication, checking that applications are authorized to access a queue or queue manager. You can also police applications' use of the API, authenticating the individual API calls, or even the parameters they use.
- For *flexibility*, you can respond to rapid changes in your business environment without changing the applications that rely on the data in that environment. You could, for example, have API exits that respond to changes in interest rates, currency exchange rates, or the price of components in a manufacturing environment.
- For *monitoring* use of a queue or queue manager, you can trace the flow of applications and messages, log errors in the API calls, set up audit trails for accounting purposes, or collect usage statistics for planning purposes.

Parent topic: API exits

This build: January 26, 2011 11:55:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19880_

7.5.2. How you use API exits

This section gives a brief overview of the tasks involved in setting up API exits.

How to configure WebSphere MQ for API exits

How to write an API exit

Parent topic: API exits

This build: January 26, 2011 11:55:56

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19890_

7.5.2.1. How to configure WebSphere MQ for API exits

You configure WebSphere® MQ to enable API exits by changing the configuration information in the usual ways:

- On WebSphere MQ for Windows, use the WebSphere MQ Explorer or the amqmdain command to make changes to configuration information within the Windows Registry.
- On WebSphere MQ for Linux (x86 platform), use the WebSphere MQ Explorer to update the WebSphere MQ configuration files, mqs.ini and qm.ini.
- On other systems directly update WebSphere MQ configuration files, mqs.ini and qm.ini.

In either case, you provide information to:

- Name the API exit
- Identify the module and entry point of the API exit code to run
- Optionally pass data with the exit
- Identify the sequence of this exit in relation to other exits

For detailed information on this configuration, see Configuring API exits. For a description

of how API exits run, see What happens when an API exit runs?.
Parent topic: How you use API exits

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19900_

7.5.2.2. How to write an API exit

This section introduces writing API exits. For detailed information, aimed at application programmers, see the WebSphere MQ Application Programming Guide.

You write your exits using the C programming language. To help you do so, we provide a sample exit, amqsaxe0, that generates trace entries to a named file. When you start writing exits, we recommend that you use this as your starting point.

Exits are available for every API call, as follows:

- MQCONN/MQCONNX, to provide a queue manager connection handle for use on subsequent API calls
- MQDISC, to disconnect from a queue manager
- MQBEGIN, to begin a global unit of work (UOW)
- MQBACK, to back out a UOW
- MQCMIT, to commit a UOW
- MQOPEN, to open an MQSeries® resource for subsequent access
- MQCLOSE, to close an MQSeries resource that had previously been opened for access
- MQGET, to retrieve a message from a queue that has previously been opened for access
- MQPUT1, to place a message on to a queue
- MQPUT, to place a message on to a queue that has previously been opened for access
- MQINQ, to inquire on the attributes of an MQSeries resource that has previously been opened for access
- MQSET, to set the attributes of a queue that has previously been opened for access

Within API exits, these calls take the general form:

```
MQ_call_EXIT (parameters)
```

where call is the API call name (PUT, GET, and so on), and the parameters control the function of the exit, primarily providing communication between the exit and the external control blocks MQAXP (the exit parameter structure) and MQAXC (the exit context structure).
Parent topic: How you use API exits

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19910_

7.5.3. What happens when an API exit runs?

The API exit routines to run are identified in stanzas on UNIX systems, and in Registry entries on Windows systems. In this section we talk about the stanzas in the configuration files mqs.ini and qm.ini, however equivalent information is available in the Registry on Windows systems, accessible through the WebSphere® MQ Explorer.

The definition of the routines can occur in three places:

1. ApiExitCommon, in the mqs.ini file, identifies routines, for the whole of WebSphere MQ, applied when queue managers start up. These can be overridden by routines defined for individual queue managers.
2. ApiExitTemplate, in the mqs.ini file, identifies routines, for the whole of WebSphere MQ, copied to the ApiExitLocal set when a new queue manager is created.
3. ApiExitLocal, in the qm.ini file, identifies routines applicable to a particular queue manager.

When a new queue manager is created, the ApiExitTemplate definitions in mqs.ini are copied to the ApiExitLocal definitions in qm.ini for the new queue manager. When a queue manager is

started, both the ApiExitCommon and ApiExitLocal definitions are used. The ApiExitLocal definitions replace the ApiExitCommon definitions if both identify a routine of the same name. The Sequence attribute, described in Attributes for all stanzas determines the order in which the routines defined in the stanzas run.

Parent topic: API exits

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19920_

7.5.4. Configuring API exits

This section tells you how to configure API exits.

Configuring API exits on UNIX systems

Configuring API exits on Windows systems

Parent topic: API exits

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19930_

7.5.4.1. Configuring API exits on UNIX systems

You define your API exits in new stanzas in the mqs.ini and qm.ini files. The sections below describe these stanzas, and the attributes within them that define the exit routines and the sequence in which they run. For guidance on the process of changing these stanzas, see Changing the configuration information.

On Linux (x86 platform) systems you can use the WebSphere® MQ Explorer to edit these entries in mqs.ini and qm.ini.

Stanzas in mqs.ini are:

ApiExitCommon

When any queue manager starts, the attributes in this stanza are read, and then overridden by the API exits defined in qm.ini.

ApiExitTemplate

When any queue manager is created, the attributes in this stanza are copied into the newly created qm.ini file under the ApiExitLocal stanza.

The stanza in qm.ini is:

ApiExitLocal

When the queue manager starts, API exits defined here override the defaults defined in mqs.ini.

Attributes for all stanzas

Sample stanzas

Changing the configuration information

Parent topic: Configuring API exits

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:

fa19940_

7.5.4.1.1. Attributes for all stanzas

All these stanzas have the following attributes:

Name=ApiExit_name

The descriptive name of the API exit passed to it in the ExitInfoName field of the MQAXP structure.

This name must be unique, no longer than 48 characters, and contain only valid characters for the names of WebSphere® MQ objects (for example, queue names).

Function=function_name

The name of the function entry point into the module containing the API exit code. This entry point is the MQ_INIT_EXIT function.

The length of this field is limited to MQ_EXIT_NAME_LENGTH.

Module=module_name

The module containing the API exit code.

If this field contains the full path name of the module it is used as is.

If this field contains just the module name, the module is located using the ExitsDefaultPath attribute in the ExitPath in qm.ini.

On platforms that support separate threaded libraries (AIX®, HP/UX, and Linux), you must provide both a non-threaded and a threaded version of the API exit module. The threaded version must have an _r suffix. The threaded version of the WebSphere MQ application stub implicitly appends _r to the given module name before it is loaded.

The length of this field is limited to the maximum path length the platform supports.

Data=data_name

Data to be passed to the API exit in the ExitData field of the MQAXP structure.

If you include this attribute, leading and trailing blanks are removed, the remaining string is truncated to 32 characters, and the result is passed to the exit. If you omit this attribute, the default value of 32 blanks is passed to the exit.

The maximum length of this field is 32 characters.

Sequence=sequence_number

The sequence in which this API exit is called relative to other API exits. An exit with a low sequence number is called before an exit with a higher sequence number. There is no need for the sequence numbering of exits to be contiguous; a sequence of 1, 2, 3 has the same result as a sequence of 7, 42, 1096. If two exits have the same sequence number, the queue manager decides which one to call first. You can tell which was called *after* the event by putting the time or a marker in ExitChainArea indicated by the ExitChainAreaPtr in MQAXP or by writing your own log file.

This attribute is an unsigned numeric value.

Parent topic: Configuring API exits on UNIX systems

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19950_

7.5.4.1.2. Sample stanzas

The mqs.ini file below contains the following stanzas:

ApiExitTemplate

This stanza defines an exit with the descriptive name OurPayrollQueueAuditor, module name auditor, and sequence number 2. A data value of 123 is passed to the exit.

ApiExitCommon

This stanza defines an exit with the descriptive name MQPoliceman, module name tmqp, and sequence number 1. The data passed is an instruction (CheckEverything).

```
mqs.ini

    ApiExitTemplate:
       Name=OurPayrollQueueAuditor
       Sequence=2
       Function=EntryPoint
       Module=/usr/ABC/auditor
       Data=123
    ApiExitCommon:
       Name=MQPoliceman
       Sequence=1
       Function=EntryPoint
       Module=/usr/MQPolice/tmqp
       Data=CheckEverything
```

The qm.ini file below contains an ApiExitLocal definition of an exit with the descriptive name ClientApplicationAPIchecker, module name ClientAppChecker, and sequence number 3.

```
qm.ini

    ApiExitLocal:
       Name=ClientApplicationAPIchecker
       Sequence=3
       Function=EntryPoint
       Module=/usr/Dev/ClientAppChecker
       Data=9.20.176.20
```

Parent topic: Configuring API exits on UNIX systems

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19960_

7.5.4.1.3. Changing the configuration information

The WebSphere® MQ configuration file, mqs.ini, contains information relevant to all the queue managers on a particular node. You can find it in the /var/mqm directory.

A queue manager configuration file, qm.ini, contains information relevant to a specific queue manager. There is one queue manager configuration file for each queue manager, held in the root of the directory tree occupied by the queue manager. For example, the path and the name for a configuration file for a queue manager called QMNAME is:

```
/var/mqm/qmgrs/QMNAME/qm.ini
```

Before editing a configuration file, back it up so that you have a copy you can revert to if the need arises.

You can edit configuration files either:

- Automatically, using commands that change the configuration of queue managers on the node
- Manually, using a standard text editor

If you set an incorrect value on a configuration file attribute, the value is ignored and an operator message is issued to indicate the problem. (The effect is the same as missing out the attribute entirely.)

Parent topic: Configuring API exits on UNIX systems

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19970_

7.5.4.2. Configuring API exits on Windows systems

You configure API exits on Windows systems using the WebSphere® MQ Explorer or the amqmdain command to update the Windows Registry.

When using the WebSphere MQ Explorer you can modify the ApiExitCommon, ApiExitTemplate and

ApiExitLocal stanzas as follows:

- The ApiExitCommon and ApiExitTemplate stanza are defined on the WebSphere MQ properties page, under Exits.
- The ApiExitLocal stanza is defined on the queue manager properties page, under Exits.

When entering, or changing, the attributes for an exit, the attributes are those defined in Attributes for all stanzas.

Parent topic: Configuring API exits

This build: January 26, 2011 11:55:57

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19980_

7.6. API exit reference information

This chapter provides reference information for the API exit. It includes:

- Data structures used by an API exit function:
  - MQACH - API exit chain header
  - MQAXC - API exit context
  - MQAXP - API exit parameter
  - MQXEPO - Register entry point options
- Calls an API exit function can issue:
  - MQXEP - Register entry point
- Definitions of the API exit functions:
  - MQ_BACK_EXIT - Back out changes
  - MQ_BEGIN_EXIT - Begin unit of work
  - MQ_CLOSE_EXIT - Close object
  - MQ_CMIT_EXIT - Commit changes
  - MQ_CONNX_EXIT - Connect queue manager (extended)
  - MQ_DISC_EXIT - Disconnect queue manager
  - MQ_GET_EXIT - Get message
  - MQ_INIT_EXIT - Initialize exit environment
  - MQ_INQ_EXIT - Inquire object attributes
  - MQ_OPEN_EXIT - Open object
  - MQ_PUT_EXIT - Put message
  - MQ_PUT1_EXIT - Put one message
  - MQ_SET_EXIT - Set object attributes
  - MQ_TERM_EXIT - Terminate exit environment

The data structures, calls, and exits are described in the order shown above (alphabetic order within each type).

General usage notes

MQACH - API exit chain header
The MQACH structure describes the header information that must be present at the start of each exit chain area.

MQAXC - API exit context

MQAXP - API exit parameter

MQXEP - Register entry point

MQXEPO - Register entry point options
The following table summarizes the fields in the structure.

MQNXP structure - Set by the caller of the exit
The MQNXP structure is allocated and managed by the caller of the exit. The structure fields are allocated by the caller of the exit or within the exit itself.

MQ_BACK_EXIT - Back out changes

MQ_BEGIN_EXIT - Begin unit of work

MQ_CALLBACK_EXIT - Callback

MQ_CLOSE_EXIT - Close object

MQ_CMIT_EXIT - Commit changes

MQ_CONNX_EXIT - Connect queue manager (extended)

MQ_DISC_EXIT - Disconnect queue manager

MQ_GET_EXIT - Get message

MQ_INIT_EXIT - Initialize exit environment

MQ_INQ_EXIT - Inquire object attributes

MQ_OPEN_EXIT - Open object

MQ_PUT_EXIT - Put message

MQ_PUT1_EXIT - Put one message

MQ_SET_EXIT - Set object attributes

MQ_SUB_EXIT - Register subscription

MQ_SUBRQ_EXIT - Subscription request

MQ_TERM_EXIT - Terminate exit environment

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa19990_

7.6.1. General usage notes

This section contains general usage notes that relate to all API exit functions.

1. All exit functions can issue the MQXEP call; this call is designed specifically for use from API exit functions.
2. The MQ_INIT_EXIT function cannot issue any MQ calls other than MQXEP.
3. All other exit functions can issue the following MQ calls:
   - MQBACK, MQBEGIN, MQBUFMH, MQCLOSE, MQCMIT, MQCONN, MQCONNX, MQCRTMH, MQDISC, MQDLTMH, MQDLTMP, MQGET, MQINQ, MQINQMP, MQMHBUF, MQOPEN, MQPUT, MQPUT1, MQSET, MQSETMP.
4. If an exit function issues the MQCONN call, or the MQCONNX call with the MQCNO_HANDLE_SHARE_NONE option, the call completes with reason code MQRC_ALREADY_CONNECTED, and the handle returned is the same as the one passed to the exit as a parameter.
5. In general when an API exit function issues an MQI call, API exits are not be called recursively. However, if an exit function issues the MQCONNX call with the MQCNO_HANDLE_SHARE_BLOCK or MQCNO_HANDLE_SHARE_NO_BLOCK options, the call returns a new shared handle. This provides the exit suite with a connection handle of its own, and

hence a unit of work that is independent of the application's unit of work. The exit suite can use this handle to put and get messages within its own unit of work, and commit or back out that unit of work; all of this can be done without affecting the application's unit of work in any way.
Because the exit function is using a connection handle that is different from the handle being used by the application, MQ calls issued by the exit function result in the relevant API exit functions being invoked. Exit functions can therefore be invoked recursively. Note that both the *ExitUserArea* field in MQAXP and the exit chain area have connection-handle scope. Consequently, an exit function cannot use those areas to signal to another instance of itself invoked recursively that it is already active.

6. Exit functions can also put and get messages within the application's unit of work. When the application commits or backs out the unit of work, all messages within the unit of work are committed or backed out together, regardless of who placed them in the unit of work (application or exit function). However, the exit can cause the application to exceed system limits sooner than would otherwise be the case (for example, by exceeding the maximum number of uncommitted messages in a unit of work).
   When an exit function uses the application's unit of work in this way, the exit function should usually avoid issuing the MQCMIT call, as this commits the application's unit of work and might impair the correct functioning of the application. However, the exit function might sometimes need to issue the MQBACK call, if the exit function encounters a serious error that prevents the unit of work being committed (for example, an error putting a message as part of the application's unit of work). When MQBACK is called, take care to ensure that the application unit of work boundaries are not changed. In this situation the exit function must set the appropriate values to ensure that completion code MQCC_WARNING and reason code MQRC_BACKED_OUT are returned to the application, so that the application can detect the fact that the unit of work has been backed out.
   If an exit function uses the application's connection handle to issue MQ calls, those calls do not themselves result in further invocations of API exit functions.
7. If an MQXR_BEFORE exit function terminates abnormally, the queue manager might be able to recover from the failure. If it can, the queue manager continues processing as though the exit function had returned MQXCC_FAILED. If the queue manager cannot recover, the application is terminated.
8. If an MQXR_AFTER exit function terminates abnormally, the queue manager might be able to recover from the failure. If it can, the queue manager continues processing as though the exit function had returned MQXCC_FAILED. If the queue manager cannot recover, the application is terminated. Be aware that in the latter case, messages retrieved outside a unit of work are lost (this is the same situation as the application failing immediately after removing a message from the queue).
9. The MCA process performs a two phase commit.
   If an API exit intercepts an MQCMIT from a prepared MCA process and attempts to perform an action within the unit of work, then the action will fail with reason code MQRC_UOW_NOT_AVAILABLE.

Parent topic: API exit reference information

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20000_

7.6.2. MQACH – API exit chain header

The MQACH structure describes the header information that must be present at the start of each exit chain area.

- The address of the first area in the chain in given by the *ExitChainAreaPtr* field in MQAXP. If there is no chain, *ExitChainAreaPtr* is the null pointer.
- The address of the next area in the chain is given by the *NextChainAreaPtr* field in MQACH. For the last area in the chain, *NextChainAreaPtr* is the null pointer.

Any exit function can create a chain area in dynamically-obtained storage (for example, by using malloc), and add that area to the chain at the required location (start, middle, or end). The exit function must ensure that it sets all fields in MQACH to valid values.

The exit suite that creates the chain area is responsible for destroying that chain area before termination (the MQ_TERM_EXIT function is a convenient point at which to do this). However, adding and removing chain areas from the chain must be done only by an exit function when it is invoked by the queue manager; this restriction is necessary to avoid serialization problems.

Exit chain areas are made available to all exit suites, and must not be used to hold private data. Use *ExitUserArea* in MQAXP to hold private data.

In general there is no correspondence between the chain of exit functions that are invoked for an API call, and the chain of exit chain areas:

- Some exit functions might not have chain areas.
- Other exit functions might each have multiple chain areas.
- The order of the chain areas might be different from the order of the exit functions that own those chain areas.

Fields

C declaration

Parent topic: API exit reference information

This build: January 26, 2011 11:55:57

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20010_

7.6.2.1. Fields

The MQACH structure contains the following fields:

StrucId (MQCHAR4)

Version (MQLONG)

StrucLength (MQLONG)

ChainAreaLength (MQLONG)

ExitInfoName (MQCHAR48)

NextChainAreaPtr (PMQACH)

Parent topic: MQACH - API exit chain header

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20020_

7.6.2.1.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQACH_STRUC_ID
  Identifier for API exit chain header structure.

  For the C programming language, the constant MQACH_STRUC_ID_ARRAY is also defined; this has the same value as MQACH_STRUC_ID, but is an array of characters instead of a string.

This initial value of this field is MQACH_STRUC_ID.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20030_

7.6.2.1.2. Version (MQLONG)

Structure version number.

The value is:

MQACH_VERSION_1
  Version-1 API exit chain header structure.

The following constant specifies the version number of the current version:

MQACH_CURRENT_VERSION
  Current version of API exit chain header structure.

Note: When a new version of the MQACH structure is introduced, the layout of the existing part is not changed. The exit function must therefore check that the version number is equal to or greater than the lowest version that contains the fields that the exit function needs to use.

The initial value of this field is MQACH_CURRENT_VERSION.

Parent topic: Fields

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20040_

7.6.2.1.3. StrucLength (MQLONG)

Length of MQACH structure.

This is the length of the MQACH structure itself; this length *excludes* the exit-defined data that follows the MQACH structure (see the *ChainAreaLength* field).

- The exit function that creates the MQACH structure must set this field to the length of the MQACH.
- An exit function that wants to access the exit-defined data should use *StrucLength* as the offset of the exit-defined data from the start of the MQACH structure.

The following value is defined:

MQACH_LENGTH_1
  Length of version-1 MQACH structure.

The following constant specifies the length of the current version:

MQACH_CURRENT_LENGTH
  Length of current version of exit chain area header.

The initial value of this field is MQACH_CURRENT_LENGTH.

Parent topic: Fields

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20050_

7.6.2.1.4. ChainAreaLength (MQLONG)

Total length of chain area.

This is the total length of the chain area. It is equal to the sum of the length of the MQACH plus the length of the exit-defined data that follows the MQACH.

The initial value of this field is zero.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20060_

7.6.2.1.5. ExitInfoName (MQCHAR48)

Exit information name.

This is a name that is used to identify the exit suite to which the chain area belongs.

The length of this field is given by MQ_EXIT_INFO_NAME_LENGTH. The initial value of this field is the null string in C.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20070_

7.6.2.1.6. NextChainAreaPtr (PMQACH)

Address of next MQACH structure in chain.

This is the address of the next chain area in the chain. If the current chain area is the last one in the chain, *NextChainAreaPtr* is the null pointer.

The initial value of this field is the null pointer.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20080_

7.6.2.2. C declaration

```
typedef struct tagMQACH MQACH;
struct tagMQACH {
  MQCHAR4    StrucId;            /* Structure identifier */
  MQLONG     Version;            /* Structure version number */
  MQLONG     StrucLength;        /* Length of MQACH structure */
  MQLONG     ChainAreaLength;    /* Total length of chain area */
  MQCHAR48   ExitInfoName;       /* Exit information name */
  PMQACH     NextChainAreaPtr;   /* Address of next MQACH structure in
                                    chain */
};
```

Parent topic: MQACH - API exit chain header

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20090_

7.6.3. MQAXC – API exit context

The MQAXC structure describes the context information that is passed to an API exit. The

context information relates to the environment in which the application is running.

- Fields
- C declaration

Parent topic: API exit reference information

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20100_

7.6.3.1. Fields

The MQAXC structure contains the following fields:

- StrucId (MQCHAR4)
- Version (MQLONG)
- Environment (MQLONG)
- UserId (MQCHAR12)
- SecurityId (MQBYTE40)
- ConnectionName (MQCHAR264)
- LongMCAUserIdLength (MQLONG)
- LongRemoteUserIdLength (MQLONG)
- LongMCAUserIdPtr (MQPTR)
- LongRemoteUserIdPtr (MQPTR)
- ApplName (MQCHAR28)
- ApplType (MQLONG)
- ProcessId (MQPID)
- ThreadId (MQTID)

Parent topic: MQAXC - API exit context

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20110_

7.6.3.1.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQAXC_STRUC_ID
  Identifier for API exit parameter structure.

  For the C programming language, the constant MQAXC_STRUC_ID_ARRAY is also defined; this has

the same value as MQAXC_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20120_

7.6.3.1.2. Version (MQLONG)

Structure version number.

The value is:

MQAXC_VERSION_1
  Version-1 API exit parameter structure.

The following constant specifies the version number of the current version:

MQAXC_CURRENT_VERSION
  Current version of API exit parameter structure.

Note: When a new version of the MQAXC structure is introduced, the layout of the existing part is not changed. The exit should therefore check that the version number is equal to or greater than the lowest version which contains the fields that the exit needs to use.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20130_

7.6.3.1.3. Environment (MQLONG)

Environment.

This indicates the environment from which the API call was issued. The value is one of the following:

MQXE_COMMAND_SERVER
  Command server.
MQXE_MQSC
  The "runmqsc" command interpreter.
MQXE_MCA
  Message channel agent.
MQXE_MCA_SVRCONN
  Message channel agent acting on behalf of a client.
MQXE_OTHER
  Environment not defined.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20140_

7.6.3.1.4. UserId (MQCHAR12)

User identifier.

This is the user identifier associated with the program that issued the API call. For a client connection (MQXE_MCA_SVRCONN), *UserId* contains the user identifier of the adopted user, and not the user identifier of the MCA.

The length of this field is given by MQ_USER_ID_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20150_

7.6.3.1.5. SecurityId (MQBYTE40)

Security identifier.

This is the security identifier associated with the program that issued the API call. For a client connection (MQXE_MCA_SVRCONN), *SecurityId* contains the security identifier of the adopted user, and not the security identifier of the MCA. If the security identifier is not known, *SecurityId* has the value:

MQSID_NONE

No security identifier specified.

The value is binary zero for the length of the field.

For the C programming language, the constant MQSID_NONE_ARRAY is also defined; this has the same value as MQSID_NONE, but is an array of characters instead of a string.

The length of this field is given by MQ_SECURITY_ID_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20160_

7.6.3.1.6. ConnectionName (MQCHAR264)

Connection name.

For a client connection (MQXE_MCA_SVRCONN), this field contains the address of the client (for example, the TCP/IP address). In other cases, this field is blank.

The length of this field is given by MQ_CONN_NAME_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20170_

7.6.3.1.7. LongMCAUserIdLength (MQLONG)

Length of long MCA user identifier.

For MQXE_MCA and MQXE_MCA_SVRCONN, this is the length in bytes of the full MCA user identifier pointed to by *LongMCAUserIdPtr*. In other cases, this field is zero.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20180_

7.6.3.1.8. LongRemoteUserIdLength (MQLONG)

Length of long remote user identifier.

For MQXE_MCA and MQXE_MCA_SVRCONN, this is the length in bytes of the full remote user identifier pointed to by *LongRemoteUserIdPtr*. In other cases, this field is zero.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20190_

7.6.3.1.9. LongMCAUserIdPtr (MQPTR)

Address of long MCA user identifier.

For MQXE_MCA and MQXE_MCA_SVRCONN, this is the address of the full MCA user identifier. The length of the full identifier is given by *LongMCAUserIdLength*. In other cases, this field is the null pointer.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20200_

7.6.3.1.10. LongRemoteUserIdPtr (MQPTR)

Address of long remote user identifier.

For MQXE_MCA and MQXE_MCA_SVRCONN, this is the address of the full remote user identifier. The length of the full identifier is given by *LongRemoteUserIdLength*. In other cases, this field is the null pointer.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20210_

7.6.3.1.11. ApplName (MQCHAR28)

Application name.

This is the name of the application that issued the API call. This name is obtained in the

same way as the default value for the *PutApplName* field in MQMD.

The length of this field is given by MQ_APPL_NAME_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20220_

7.6.3.1.12. ApplType (MQLONG)

Application type.

This is the type of the application that issued the API call. The value is the same as MQAT_DEFAULT for the environment for which the application was compiled.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20230_

7.6.3.1.13. ProcessId (MQPID)

The WebSphere® MQ process identifier.

This is the same identifier used in WebSphere MQ trace and FFST™ dumps, but might be different from the operating system process identifier. Where applicable, the exit handler sets this field on entry to each exit function.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20240_

7.6.3.1.14. ThreadId (MQTID)

The WebSphere® MQ thread identifier.

This is the same identifier used in WebSphere MQ trace and FFST™ dumps, but might be different from the operating system thread identifier. Where applicable, the exit handler sets this field on entry to each exit function.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20250_

7.6.3.2. C declaration

```
typedef struct tagMQAXC MQAXC;
```

```
struct tagMQAXC {
  MQCHAR4     StrucId;                  /* Structure identifier */
  MQLONG      Version;                  /* Structure version number */
  MQLONG      Environment;              /* Environment */
  MQCHAR12    UserId;                   /* User identifier */
  MQBYTE40    SecurityId;               /* Security identifier */
  MQCHAR264   ConnectionName;           /* Connection name */
  MQLONG      LongMCAUserIdLength;      /* Length of long MCA user
                                           identifier */
  MQLONG      LongRemoteUserIdLength;   /* Length of long remote user
                                           identifier */
  MQPTR       LongMCAUserIdPtr;         /* Address of long MCA user
                                           identifier */
  MQPTR       LongRemoteUserIdPtr;      /* Address of long remote user
                                           identifier */
  MQCHAR28    ApplName;                 /* Application name */
  MQLONG      ApplType;                 /* Application type */
  MQPID       ProcessId;                /* Process identifier */
  MQTID       ThreadId;                 /* Thread identifier */
};
```

Parent topic: MQAXC - API exit context

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20260_

7.6.4. MQAXP – API exit parameter

The MQAXP structure describes the information that is passed to an API exit.

Fields

C declaration

Parent topic: API exit reference information

This build: January 26, 2011 11:55:58

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20270_

7.6.4.1. Fields

The MQAXP structure contains the following fields:

StrucId (MQCHAR4)

Version (MQLONG)

ExitId (MQLONG)

ExitReason (MQLONG)

ExitResponse (MQLONG)

ExitResponse2 (MQLONG)

Feedback (MQLONG)

APICallerType (MQLONG)

ExitUserArea (MQBYTE16)

ExitData (MQCHAR32)

ExitInfoName (MQCHAR48)

ExitPDArea (MQBYTE48)

QMgrName (MQCHAR48)

ExitChainAreaPtr (PMQACH)

Hconfig (MQHCONFIG)

Function (MQLONG)

ExitMsgHandle (MQHMSG)

Parent topic: MQAXP - API exit parameter

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20280_

7.6.4.1.1. StrucId (MQCHAR4)

Structure identifier.

The value is:

MQAXP_STRUC_ID
  Identifier for API exit parameter structure.

  For the C programming language, the constant MQAXP_STRUC_ID_ARRAY is also defined; this has the same value as MQAXP_STRUC_ID, but is an array of characters instead of a string.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20290_

7.6.4.1.2. Version (MQLONG)

Structure version number.

The value is:

MQAXP_VERSION_1
  Version-1 API exit parameter structure.

The following constant specifies the version number of the current version:

MQAXP_CURRENT_VERSION
  Current version of API exit parameter structure.

Note: When a new version of the MQAXP structure is introduced, the layout of the existing part is not changed. The exit should therefore check that the version number is equal to or greater than the lowest version which contains the fields that the exit needs to use.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20300_

7.6.4.1.3. ExitId (MQLONG)

Type of exit.

This indicates the type of exit being called. The value is:

MQXT_API_EXIT

API exit.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:58

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20310_

7.6.4.1.4. ExitReason (MQLONG)

Reason for invoking exit.

This indicates the reason why the exit is being called. Possible values are:

MQXR_CONNECTION

Connection level processing.

The exit is invoked with this value twice for each connection:

- Before the MQCONN or MQCONNX call, so that the exit can perform connection-level initialization. The *Function* field has the value MQXF_INIT in this case.
  The MQXF_INIT exit function should be used for general initialization of the exit suite, and the MQXF_CONN or MQXF_CONNX exit functions should be used specifically for processing the MQCONN or MQCONNX calls.
- After the MQDISC call, so that the exit can perform connection-level termination. The *Function* field has the value MQXF_TERM in this case.
  The MQXF_TERM exit function should be used for general termination of the exit suite, and the MQXF_DISC exit function should be used specifically for processing the MQDISC call.

MQXR_BEFORE

Before API execution.

The *Function* field can have any of the MQXF_* values other than MQXF_INIT or MQXF_TERM.

For the MQGET call, this value occurs with the:

- MQXF_GET exit function before API execution
- MQXF_DATA_CONV_ON_GET exit function after API execution but before data conversion

MQXR_AFTER

After API execution.

The *Function* field can have any of the MQXF_* values other than MQXF_INIT, MQXF_TERM, or MQXF_DATA_CONV_ON_GET.

For the MQGET call, this value occurs with the:

- MQXF_GET exit function after both API execution and data conversion have been completed

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20320_

7.6.4.1.5. ExitResponse (MQLONG)

Response from exit.

This is set by the exit function to indicate the outcome of the processing performed by the exit. It must be one of the following:

MQXCC_OK

Exit completed successfully.

This value can be set by all MQXR_* exit functions. The *ExitResponse2* field must be set by the exit function to indicate how processing should continue.

Note: Returning MQXCC_OK does *not* imply that the completion code for the API call is MQCC_OK, or that the reason code is MQRC_NONE.

MQXCC_FAILED

Exit failed.

This value can be set by all MQXR_* exit functions. It causes the queue manager to set the completion code for the API call to MQCC_FAILED, and the reason code to one of the following values:

Exit function	Reason code set by queue manager
MQXF_INIT	MQRC_API_EXIT_INIT_ERROR
MQXF_TERM	MQRC_API_EXIT_TERM_ERROR
All others	MQRC_API_EXIT_ERROR

However, the values set by the queue manager can be altered by an exit function later in the chain.

The *ExitResponse2* field is ignored; the queue manager continues processing as though MQXR2_SUPPRESS_CHAIN had been returned:

- **For an MQXR_BEFORE exit function, processing continues with the MQXR_AFTER exit function that matches this MQXR_BEFORE exit function (that is, all intervening MQXR_BEFORE and MQXR_AFTER exit functions, plus the API call itself, are skipped).**
- **For an MQXR_AFTER exit function, processing continues with the next MQXR_AFTER exit function in the chain.**

MQXCC_SUPPRESS_FUNCTION

Suppress function.

If an MQXR_BEFORE exit function returns this value, the queue manager sets the completion code for the API call to MQCC_FAILED, the reason code to MQRC_SUPPRESSED_BY_EXIT, and the API call is skipped. If returned by the MQXF_DATA_CONV_ON_GET exit function, data conversion is skipped.

The *ExitResponse2* field must be set by the exit function to indicate whether the remaining MQXR_BEFORE exit functions and their matching MQXR_AFTER exit functions should be invoked. Any of these exit functions can alter the completion code and reason code of the API call that were set by the queue manager.

If an MQXR_AFTER or MQXR_CONNECTION exit function returns this value, the queue manager continues processing as though the exit had returned MQXCC_FAILED.

MQXCC_SKIP_FUNCTION

Skip function.

This is the same as MQXCC_SUPPRESS_FUNCTION, except the exit function can set the completion code and reason code of the API call.

MQXCC_SUPPRESS_EXIT

Suppress exit.

If an MQXR_BEFORE or MQXR_AFTER exit function returns this value, the queue manager deregisters immediately all of the exit functions belonging to this exit suite. The only exception is the MQXF_TERM exit function, which will be invoked at termination of the

connection if registered when MQXCC_SUPPRESS_EXIT is returned. Note that if an MQXR_BEFORE exit function returns this value, the matching MQXR_AFTER exit function will *not* be invoked after the API call, since that exit function will no longer be registered.

The *ExitResponse2* field must be set by the exit function to indicate whether the remaining MQXR_BEFORE exit functions and their matching MQXR_AFTER exit functions should be invoked.

If an MQXR_CONNECTION exit function returns this value, the queue manager continues processing as though the exit had returned MQXCC_FAILED.

If the exit function sets *ExitResponse* to a value that is not valid, the queue manager continues processing as though the exit had returned MQXCC_FAILED.

On entry to the exit function, *ExitResponse* has the value MQXCC_OK.

This is an output field from the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20330_

7.6.4.1.6. ExitResponse2 (MQLONG)

Secondary response from exit.

This is the secondary exit response code that can be set by an MQXR_BEFORE exit function to provide additional information to the queue manager. If set by an MQXR_AFTER or MQXR_CONNECTION exit function, the value is ignored. The value must be one of the following:

MQXR2_DEFAULT_CONTINUATION

Default continuation.

Continuation with the next exit function in the chain depends on the value of the *ExitResponse* field:

- If *ExitResponse* is MQXCC_OK or MQXCC_SUPPRESS_EXIT, the next MQXR_BEFORE exit function in the chain is invoked.
- If *ExitResponse* is MQXCC_SUPPRESS_FUNCTION or MQXCC_SKIP_FUNCTION, no further MQXR_BEFORE exit functions are invoked for this particular API call.

MQXR2_CONTINUE_CHAIN

Continue with next MQXR_BEFORE exit function in chain.

MQXR2_SUPPRESS_CHAIN

Skip remaining MQXR_BEFORE exit functions in chain.

All subsequent MQXR_BEFORE exit functions in the chain, and their matching MQXR_AFTER exit functions, are skipped for this particular API call. The MQXR_AFTER exit functions that match the current exit function and earlier MQXR_BEFORE exit functions are not skipped.

If the exit function sets *ExitResponse2* to a value that is not valid, the queue manager continues processing as though the exit had returned MQXR2_DEFAULT_CONTINUATION.

This is an output field from the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20340_

7.6.4.1.7. Feedback (MQLONG)

Feedback.

This is a field that allows the exit functions belonging to an exit suite to communicate feedback codes both to each other, and to exit functions belonging to other exit suites. The

field is initialized to MQFB_NONE before the first invocation of the first exit function in the first exit suite (the MQXF_INIT exit function), and thereafter any changes made to this field by exit functions are preserved across the invocations of the exit functions.

This is an input/output field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20350_

7.6.4.1.8. APICallerType (MQLONG)

API caller type.

This indicates the type of program that issued the API call that caused the exit function to be invoked. The value is one of the following:

MQXACT_EXTERNAL

Caller is external to the queue manager.

MQXACT_INTERNAL

Caller is internal to the queue manager.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20360_

7.6.4.1.9. ExitUserArea (MQBYTE16)

Exit user area.

This is a field that allows exit functions belonging to the same exit suite to share data with each other, but not with other exit suites. The field is initialized to MQXUA_NONE (binary zero) before the first invocation of the first exit function in the exit suite (the MQXF_INIT exit function), and thereafter any changes made to this field by exit functions are preserved across the invocations of the exit functions. The queue manager resets the field to MQXUA_NONE when control returns from the MQXF_TERM exit function to the queue manager.

The following value is defined:

MQXUA_NONE

No user information.

The value is binary zero for the length of the field.

For the C programming language, the constant MQXUA_NONE_ARRAY is also defined; this has the same value as MQXUA_NONE, but is an array of characters instead of a string.

The length of this field is given by MQ_EXIT_USER_AREA_LENGTH. This is an input/output field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20370_

7.6.4.1.10. ExitData (MQCHAR32)

Exit data.

On input to each exit function, this field is set to the character data associated with the definition of the exit suite to which the exit function belongs. If no value has been defined for that data, *ExitData* is blank.

The length of this field is given by MQ_EXIT_DATA_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20380_

7.6.4.1.11. ExitInfoName (MQCHAR48)

Exit information name.

This is a name that is used to identify the exit suite to which the exit function belongs.

The length of this field is given by MQ_EXIT_INFO_NAME_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20390_

7.6.4.1.12. ExitPDArea (MQBYTE48)

Problem determination area.

This is a field that is available for the exit to use, to assist with problem determination. The field is initialized to MQXPDA_NONE (binary zero) before each invocation of the exit function. The exit function can set this field to any value it chooses. When the exit returns control to the queue manager, the contents of *ExitPDArea* are written to the trace file, if tracing is active.

The following value is defined:

MQXPDA_NONE

No problem-determination information.

The value is binary zero for the length of the field.

For the C programming language, the constant MQXPDA_NONE_ARRAY is also defined; this has the same value as MQXPDA_NONE, but is an array of characters instead of a string.

The length of this field is given by MQ_EXIT_PD_AREA_LENGTH. This is an input/output field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20400_

7.6.4.1.13. QMgrName (MQCHAR48)

Name of local queue manager.

This is the name of the queue manager that invoked the exit function. *QMgrName* is never

blank.

The length of this field is given by MQ_Q_MGR_NAME_LENGTH. This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20410_

7.6.4.1.14. ExitChainAreaPtr (PMQACH)

Address of first MQACH structure in chain.

The exit chain area allows exit functions belonging to one exit suite to share data with exit functions belonging to another exit suite. The exit chain area is a chain of MQACH structures that is made available to all exit functions. The address of the first MQACH structure in the chain is passed to each exit function in the *ExitChainAreaPtr* field. The exit function can scan the chain, and examine or alter the data contained within it. However, this should be done only with the prior agreement of the owner of the data.

If there is no current exit chain area, *ExitChainAreaPtr* is the NULL pointer. An exit function can at any time create an MQACH structure in storage obtained dynamically (for example, by using the C function malloc), and add it to the chain. The exit suite which creates an MQACH is responsible for freeing the storage associated with the MQACH before the exit suite terminates.

If data is to be shared between different exit functions belonging to the same exit suite, but that data is *not* to be made available to other exit suites, the *ExitUserArea* field should be used in preference to *ExitChainAreaPtr*.

This is an input/output field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20420_

7.6.4.1.15. Hconfig (MQHCONFIG)

Configuration handle.

This handle represents the set of exit functions that belong to the exit suite whose name is given by the *ExitInfoName* field. The queue manager generates a new configuration handle when the MQXF_INIT exit function is invoked, and passes that handle to the other exit functions that belong to the exit suite. This handle must be specified on the MQXEP call in order to register the entry point for an exit function.

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20430_

7.6.4.1.16. Function (MQLONG)

API function identifier.

This is the identifier of the API call that is about to be executed (when *ExitReason* has the value MQXR_BEFORE), or the API call that has just been executed (when *ExitReason* has the value MQXR_AFTER). If *ExitReason* has the value MQXR_CONNECTION, *Function* indicates whether

the exit should perform initialization or termination. The value is one of the following:

MQXF_INIT
  Initialization of exit suite.
MQXF_TERM
  Termination of exit suite.
MQXF_CONN
  MQCONN call.
MQXF_CONNX
  MQCONNX call.
MQXF_DISC
  MQDISC call.
MQXF_OPEN
  MQOPEN call.
MQXF_CLOSE
  MQCLOSE call.
MQXF_PUT1
  MQPUT1 call.
MQXF_PUT
  MQPUT call.
MQXF_GET
  MQGET call.
MQXF_DATA_CONV_ON_GET
  Data conversion on MQGET call.
MQXF_INQ
  MQINQ call.
MQXF_SET
  MQSET call.
MQXF_BEGIN
  MQBEGIN call.
MQXF_CMIT
  MQCMIT call.
MQXF_BACK
  MQBACK call.
MQXF_SUB
  MQSUB call
MQXF_SUBRQ
  MQSUBRQ call

This is an input field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20440_

7.6.4.1.17. ExitMsgHandle (MQHMSG)

When *Function* is MQXF_GET and *ExitReason* is MQXR_AFTER, a valid message handle is returned in this field allowing the API exit access to the message descriptor fields and any other properties matching the *ExitProperties* string specified in the MQXEPO structure when registering the API exit.

Any non-message descriptor properties that are returned in the *ExitMsgHandle* will not be available from the *MsgHandle* in the MQGMO structure if one was specified, or in the message

data.

When *Function* is MQXF_GET and *ExitReason* is MQXR_BEFORE, if the exit program sets this field to MQHM_NONE then it will suppress the populating of the ExitMsgHandle properties.

This field is not set if *Version* is less then MQAXP_VERSION_2.

This is an input/output field to the exit.

Parent topic: Fields

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22810_

7.6.4.2. C declaration

```
typedef struct tagMQAXP MQAXP;
struct tagMQAXP {
    MQCHAR4     StrucId;             /* Structure identifier */
    MQLONG      Version;             /* Structure version number */
    MQLONG      ExitId;              /* Type of exit */
    MQLONG      ExitReason;          /* Reason for invoking exit */
    MQLONG      ExitResponse;        /* Response from exit */
    MQLONG      ExitResponse2;       /* Secondary response from exit */
    MQLONG      Feedback;            /* Feedback */
    MQLONG      APICallerType;       /* API caller type */
    MQBYTE16    ExitUserArea;        /* Exit user area */
    MQCHAR32    ExitData;            /* Exit data */
    MQCHAR48    ExitInfoName;        /* Exit information name */
    MQBYTE48    ExitPDArea;          /* Problem determination area */
    MQCHAR48    QMgrName;            /* Name of local queue manager */
    PMQACH      ExitChainAreaPtr;   /* Address of first MQACH
                                       structure in chain */
    MQHCONFIG   Hconfig;             /* Configuration handle */
    MQLONG      Function;            /* API function identifier */
    MQHMSG      ExitMsgHandle;       /* Exit message handle */
 };
```

Parent topic: MQAXP – API exit parameter

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20450_

7.6.5. MQXEP – Register entry point

This call is used by an exit function to register the entry points of other exit functions in the exit suite. This is usually done by the MQ_INIT_EXIT function, but can be done by any exit function in the exit suite.

The MQXEP call is also used to deregister entry points. This is usually done by the MQ_TERM_EXIT function, but can be done by any exit function in the exit suite.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20460_

7.6.5.1. Syntax

MQXEP *(Hconfig, ExitReason, Function, EntryPoint, ExitOpts, pCompCode, pReason)*

Parent topic: MQXEP – Register entry point

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20470_

7.6.5.2. Parameters

The MQXEP call has the following parameters.

- Hconfig (MQHCONFIG) – input
- ExitReason (MQLONG) – input
- Function (MQLONG) – input
- EntryPoint (PMQFUNC) – input
- ExitOpts (MQXEPO) – input
- pCompCode (PMQLONG) – output
- pReason (PMQLONG) – output

Parent topic: MQXEP – Register entry point

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20480_

7.6.5.2.1. Hconfig (MQHCONFIG) – input

Configuration handle.

This handle represents the exit suite to which the current exit function belongs. The queue manager generates this configuration handle when the MQ_INIT_EXIT function is invoked, and uses the *Hconfig* field in the MQAXP structure to pass the handle to each exit function in the exit suite.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20490_

7.6.5.2.2. ExitReason (MQLONG) – input

Exit reason.

This specifies when to call the entry point being registered or deregistered. It must be one of the following:

MQXR_CONNECTION

Connection level processing.

The *Function* parameter must have the value MQXF_INIT or MQXF_TERM.

MQXR_BEFORE

Before API execution.

The *Function* parameter can have any of the MQXF_* values other than MQXF_INIT or MQXF_TERM.

MQXR_AFTER

After API execution.

The *Function* parameter can have any of the MQXF_* values other than MQXF_INIT, MQXF_TERM, or MQXF_DATA_CONV_ON_GET.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20500_

7.6.5.2.3. Function (MQLONG) – input

Function identifier.

This specifies the API call for which the entry point is being registered or deregistered. It must be one of the following:

MQXF_INIT

Initialization of exit suite.

MQXF_TERM

Termination of exit suite.

MQXF_CONN

MQCONN call.

MQXF_CONNX

MQCONNX call.

MQXF_DISC

MQDISC call.

MQXF_OPEN

MQOPEN call.

MQXF_CLOSE

MQCLOSE call.

MQXF_PUT1

MQPUT1 call.

MQXF_PUT

MQPUT call.

MQXF_GET

MQGET call.

MQXF_DATA_CONV_ON_GET

Data conversion on MQGET call.

MQXF_INQ

MQINQ call.

MQXF_SET

MQSET call.

MQXF_BEGIN

MQBEGIN call.

MQXF_CMIT

MQCMIT call.

MQXF_BACK

MQBACK call.

If the MQXEP call is used more than once to register different entry points for a particular combination of *Function* and *ExitReason*, the last call made provides the entry point that is used.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20510_

7.6.5.2.4. EntryPoint (PMQFUNC) – input

Exit function entry point.

This is the address of the entry point being registered.

If the value specified is the null pointer, it indicates either that the exit function is not provided, or that a previously-registered exit function is being deregistered. The null pointer is assumed for entry points which are not defined using MQXEP.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20520_

7.6.5.2.5. ExitOpts (MQXEPO) – input

Exit reason.

The MQXEPO structure allows API exits to register for specific properties to be returned in the *ExitMsgHandle* of the MQAXP structure.

If a null pointer is specified for this field then the default values of the MQXEPO structure are assumed.

See MQXEPO - Register entry point options for details.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22800_

7.6.5.2.6. pCompCode (PMQLONG) – output

Completion code.

The value returned is one of the following:

MQCC_OK

Successful completion.

MQCC_FAILED

Call failed.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20540_

7.6.5.2.7. pReason (PMQLONG) – output

Reason code qualifying *pCompCode*.

If *CompCode* is MQCC_OK:

MQRC_NONE

(0, X'000') No reason to report.

If *CompCode* is MQCC_FAILED:

MQRC_EXIT_REASON_ERROR

(2377, X'949') Exit reason not valid.

MQRC_FUNCTION_ERROR

(2281, X'8E9') Function identifier not valid.

MQRC_HCONFIG_ERROR

(2280, X'8E8') Configuration handle not valid.

MQRC_PROPERTY_NAME_ERROR

(2442, X'098A') Exit properties field is not valid.

MQRC_RESERVED_VALUE_ERROR

(2378, X'94A') Reserved value not valid.

MQRC_RESOURCE_PROBLEM

(2102, X'836') Insufficient system resources available.

MQRC_UNEXPECTED_ERROR

(2195, X'893') Unexpected error occurred.

MQRC_XEPO_ERROR

(2507, X'09CB') Exit options structure not valid.

For more information on these reason codes, see the WebSphere MQ Application Programming Reference.

Parent topic: Parameters

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20550_

7.6.5.3. C invocation

```
MQXEP (Hconfig, ExitReason, Function, EntryPoint, &ExitOpts,
       &CompCode, &Reason);
```

Declare the parameters as follows:

```
MQHCONFIG Hconfig;     /* Configuration handle */
MQLONG    ExitReason; /* Exit reason */
MQLONG    Function;   /* Function identifier */
PMQFUNC   EntryPoint; /* Exit function entry point */
MQPTR     ExitOpts;   /* Exit options structure */
MQLONG    CompCode;   /* Completion code */
MQLONG    Reason;     /* Reason code qualifying CompCode */
```

Parent topic: MQXEP - Register entry point

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20560_

7.6.6. MQXEPO – Register entry point options

The following table summarizes the fields in the structure.

Field	Description	Topic
StrucId	Structure identifier	StrucId
Version	Structure version number	Version
Options	Options	Options
ExitProperties	Exit properties	ExitProperties

Overview
The MQXEPO structure allows applications to specify options that control how API exit entry points are registered.

Fields

Language declarations
This structure is supported in the following programming languages.

Parent topic: API exit reference information

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26000_

7.6.6.1. Overview

The MQXEPO structure allows applications to specify options that control how API exit entry points are registered.

The structure is an input parameter on the MQXEP call.

Parent topic: MQXEPO – Register entry point options

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26010_

7.6.6.2. Fields

The MQXEPO structure contains the following fields; the fields are described in alphabetic order:

ExitProperties (MQCHARV)
This specifies the set of properties that are made available from the *ExitMsgHandle* field of the MQAXP structure.

Options (MQLONG)

StrucId (MQCHAR4)

Version (MQLONG)

Parent topic: MQXEPO – Register entry point options

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26020_

7.6.6.2.1. ExitProperties (MQCHARV)

This specifies the set of properties that are made available from the *ExitMsgHandle* field of the MQAXP structure.

The ExitProperties name indicates the prefix for all properties belonging to the exit. For example, an ExitProperties name of "ABC" causes all properties with the prefix of "ABC." to be made available from the ExitMsgHandle field of the MQAXP structure after an MQGET call. If you include a period "." or the wildcard character "%" in the ExitProperties name the MQXEP call fails with MQRC_PROPERTY_NAME_ERROR.

Any properties added to the *ExitMsgHandle* field are not made available to the application getting the data, from the *MsgHandle* field of the MQGMO structure, if one was specified, or in the message data.

This field only applies for the exit after the MQGET call, and is ignored if the entry point being registered is not MQXF_GET for ExitReason MQXR_AFTER.

This is always an input field. The initial value of this field is MQCHARV_DEFAULT.

Parent topic: Fields

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26030_

7.6.6.2.2. Options (MQLONG)

Register entry point options structure - *Options* field.

The value must be:

MQXEPO_NONE

No options specified

This is always an input field. The initial value of this field is MQXEPO_NONE.

Parent topic: Fields

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26040_

7.6.6.2.3. StrucId (MQCHAR4)

This is the structure identifier. The value must be:

MQXEPO_STRUC_ID

Identifier for register entry point options structure.

For the C programming language, the constant QXEPO_STRUC_ID_ARRAY is also defined, and has the same value as MQXEPO _STRUC_ID. However, QXEPO_STRUC_ID_ARRAY is an array of characters instead of a string.

This is always an input field. The initial value of this field is MQXEPO_STRUC_ID.

Parent topic: Fields

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26050_

7.6.6.2.4. Version (MQLONG)

This is the structure version number. The value must be:

MQXEPO_VERSION_1
  Version-1 register entry point options structure.

The following constant specifies the version number of the current version:

MQXEPO_CURRENT_VERSION
  Current version of register entry point options structure.

This is always an input field. The initial value of this field is MQXEPO_VERSION_1.

Parent topic: Fields

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26060_

7.6.6.3. Language declarations

This structure is supported in the following programming languages.

C declaration

Parent topic: MQXEPO - Register entry point options

This build: January 26, 2011 11:56:24

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26070_

7.6.6.3.1. C declaration

```
typedef struct tagMQXEPO MQXEPO;
struct tagMQXEPO {
  MQCHAR4   StrucId;          /* Structure identifier */
  MQLONG    Version;          /* Structure version number */
  MQLONG    Options;          /* Options that control the
                                 action of MQXEP */
  MQCHARV   ExitProperties;   /* Exit properties */
};
```

Parent topic: Language declarations

This build: January 26, 2011 11:56:25

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fr26080_

7.6.7. MQNXP structure - Set by the caller of the exit

The MQNXP structure is allocated and managed by the caller of the exit. The structure fields are allocated by the caller of the exit or within the exit itself.

Table 1. Fields in MQNXP

Field	Description
StrucID	Structure identifier
Version	Structure version number
ExitId	Type of exit
ExitReason	Reason for starting exit

ExitResponse	Response from exit
ExitResponse2	Secondary response from exit
Feedback	Feedback code (reserved)
pExitUserAreaPtr	Exit user area
pExitDataPtr	Exit data
ExitDataLength	Exit data length
ppMQCDArrayPtr	Array of pointers to MQCDs
MQCDArrayCount	Number of entries found
MaxMQCDVersion	Maximum MQCD version

Fields

StrucID

Type: MQCHAR4 - input

Structure identifier. The value is as follows:

MQNXP_STRUC_ID

Structure identifier set by the caller of the exit.

For the C programming language, the constant MQNXP_STRUC_ID_ARRAY is also defined. This identifier has the same value as MQNXP_STRUC_ID, but is an array of characters instead of a string.

This field is validated when the exit is called.

Version

Type: MQLONG - input

Structure version number. The value is as follows:

MQNXP_VERSION_1

Version 1 Structure version number which id set by the caller of the exit.

The following constant specifies the version number of the current version:

MQNXP_CURRENT_VERSION

Current version of the structure, which is managed and allocated by the caller of the Exit.

The initial value of the *Version* field is MQNXP_CURRENT_VERSION_1.

ExitId

Type: MQLONG - input

Type of exit. This field is set by the caller of the exit and indicates the type of exit being called. The value is as follows:

MQXT_PRECONNECT_EXIT

API exit

ExitReason

Type: MQLONG - input

Reason for starting the exit. This field is set by the caller of the exit and indicates the reason why the exit is being called. The exit can be called for one of the following reasons:

MQXR_INIT

The exit is being initialized.

This value indicates that the exit is being started for the first time. It provides the option to the exit to acquire and initialize any resources that it needs; memory, for example.

MQXR_PRECONNECT

The exit is being started before a connection is being made.

All connection parameters can be changed.

MQXR_TERM

The exit is ended.

This field indicates that the exit is about to be stopped. The exit frees any resources that it has acquired since it was initialized; memory, for example.

ExitResponse

Type: MQLONG - output

Response from exit. This field is set to MQXCC_OK on entry to the exit routine. The field is set by the exit function to indicate the outcome of the processing performed by the exit. It must be one of the following values:

MQXCC_OK

Exit completed successfully.

The *ExitResponse2* field must be set by the exit function to indicate how processing might continue.

MQXCC_SUPPRESS_FUNCTION

Suppresses the function.

If an exit returns this value, the completion code is set to MQCC_FAILED, and the reason code is set to MQRC_SUPPRESSED_BY_EXIT. The MQCONN(X) is not issued.

The *ExitResponse2* field is set by the exit function to indicate whether the remaining exit functions in the chain must be started. Any of these exit functions can alter the completion code and the reason code that have been set.

MQXCC_FAILED

Exit has failed.

The *ExitResponse2* field is ignored; the client continues processing as though MQXR2_SUPPRESS_CHAIN had been returned.

ExitResponse2

Type: MQLONG - output

Secondary response from exit - This field is set to MQXR2_DEFAULT_CONTINUATION on entry to the exit routine. It can be set by the exit to provide further information to the client. The *ExitResponse2* field must be one of the following values:

MQXR2_DEFAULT_CONTINUATION

Default continuation.

Continuation with the next exit in the chain depends on the response from the last exit started:

If MQXCC_SUPPRESS_FUNCTION was returned, no further exits in the chain are called. Else, the next exit in the chain is started.

MQXR2_CONTINUE_CHAIN

Continue with the next exit.

MQXR2_SUPPRESS_CHAIN

Skips remaining exits in the chain.

Feedback

Type: MQLONG - output

This field specifies the feedback code that is reserved for future use and is set by the caller of the exit. This field must be set to MQFB_NONE.

pExitUserAreaPtr

Type: MQPTR - output

This field specifies the exit user area and is available for the exit to use. Any changes made to this field by the exit are preserved across invocations of the exit. It is expected that the exit uses this field to contain a pointer to a memory block allocated and maintained by the PreConnect exit itself.

pExitDataPtr

Type: PMQCHAR - input

This field comprises input data passed into the exit. The memory is allocated and maintained by the caller of the exit. The memory block is allocated on entry to the exit

routine. It is populated with the information provided by the WebSphere MQ Administrator in the data field in the PreConnect stanza of the mqclient.ini file, before an MQXR_INIT exit. It is released after an MQXR_TERM exit. If no such information is available, *ExitDataLength* is zero.

ExitDataLength

Type: MQLONG - input

The length of the data pointed to by the *pExitDataPtr* field. The field is set by the caller of the exit.

ppMQCDArrayPtr

Type: PMQCD * - input/output

Array of pointers to MQCDs. The field contains a pointer to a memory block allocated and maintained by the PreConnect exit itself. The memory block contains an array of pointers to more memory blocks containing MQCDs. Both the pointer array and the MQCD that each entry references are allocated and maintained by the PreConnect exit.

An MQCD supplied by the application through the MQCNO structure takes precedence over any connection definitions that are part of the MQXR_PRECONNECT exit. In that case, this field is ignored by the exit interface.

An exit started with MQXR_TERM releases the memory allocated for the pointer array and for every MQCD that the array references.

The interface guarantees on an exit-by-exit basis. The data referenced through this field when the exit is starting (ExitReason=MQXR_PRECONNECT) is the same as when it is ending (ExitReason=MQXR_TERM). Each exit therefore maintains its own data. If you want an exit to append to the array during MQXR_PRECONNECT, copy the existing array into a new memory block owned by the current exit.

MQCDArrayCount

Type: MQLONG - input/output

Specifies the number of MQCD entries. The field contains the number of channel definitions found as a result of the search (the number of MQCD entries reference through the *ppMQCDArrayPtr* field).

MaxMQCDVersion

Type: MQLONG - input

On input, this field states the highest supported MQCD version that the exit can use when creating the array of MQCDs.

C Declaration

```
typedef struct tagMQNXP MQNXP;
typedef MQNXP MQPOINTER PMQNXP;

struct tagMQNXP
 MQCHAR4   StrucId;                              /structure identifier */
  MQLONG   Version;                              /* Structure version number */
  MQLONG   ExitId;                             /* Type of exit */
  MQLONG   ExitReason;                         /* Reason for invoking exit */
  MQLONG   ExitResponse;                  /* Response from exit */
  MQLONG   ExitResponse2;                 /* Secondary response from exit */
  MQLONG   Feedback;                      /* Feedback code (reserved) */
  MQPTR          pExitUserAreaPtr; /* Exit user area */
  PMQCHAR        pExitDataPtr;      /* Exit data */
  MQLONG   ExitDataLength;           /* Exit data length */
  PMQCD *  ppMQCDArrayPtr;    /* Array of pointers to MQCDs */
  MQLONG   MQCDArrayCount;           /* Number of entries found */
  MQLONG   MaxMQCDVersion;           /* Maximum MQCD version
};
```

Parent topic: API exit reference information

This build: January 26, 2011 11:56:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa24010_

7.6.8. MQ_BACK_EXIT – Back out changes

Exit providers can supply an MQ_BACK_EXIT function to intercept the MQBACK call. If the unit of work is being coordinated by an external unit-of-work manager, MQ_BACK_EXIT is also invoked in response to the application issuing the unit-of-work manager's back-out call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20570_

7.6.8.1. Syntax

MQ_BACK_EXIT *(pExitParms, pExitContext, pHconn, pCompCode, pReason)*

Parent topic: MQ_BACK_EXIT – Back out changes

This build: January 26, 2011 11:55:59

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20580_

7.6.8.2. Parameters

The MQ_BACK_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

pHconn (PMQHCONN) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_BACK_EXIT – Back out changes

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20590_

7.6.8.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20600_

7.6.8.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20610_

7.6.8.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20620_

7.6.8.2.4. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20630_

7.6.8.2.5. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20640_

7.6.8.3. C invocation

```
MQ_BACK_EXIT (&ExitParms, &ExitContext, &Hconn,
              &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_BACK_EXIT - Back out changes

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20650_

7.6.9. MQ_BEGIN_EXIT – Begin unit of work

Exit providers can supply an MQ_BEGIN_EXIT function to intercept the MQBEGIN call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20660_

7.6.9.1. Syntax

MQ_BEGIN_EXIT *(pExitParms, pExitContext, pHconn, ppBeginOptions, pCompCode, pReason)*

Parent topic: MQ_BEGIN_EXIT – Begin unit of work

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20670_

7.6.9.2. Parameters

The MQ_BEGIN_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

pHconn (PMQHCONN) – input/output

ppBeginOptions (PPMQBO) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_BEGIN_EXIT – Begin unit of work

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20680_

7.6.9.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20690_

7.6.9.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20700_

7.6.9.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20710_

7.6.9.2.4. ppBeginOptions (PPMQBO) – input/output

Options that control the action of MQBEGIN.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20720_

7.6.9.2.5. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20730_

7.6.9.2.6. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa20740_

7.6.9.3. C invocation

```
MQ_BEGIN_EXIT (&ExitParms, &ExitContext, &Hconn,
               &pBeginOptions, &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;       /* Exit parameter structure */
PMQAXC     pExitContext;     /* Exit context structure */
PMQHCONN   pHconn;           /* Connection handle */
PPMQBO     ppBeginOptions;   /* Options that control the action of
                                MQBEGIN */
PMQLONG    pCompCode;        /* Completion code */
PMQLONG    pReason;          /* Reason code qualifying CompCode */
```

Parent topic: MQ_BEGIN_EXIT - Begin unit of work

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20750_

7.6.10. MQ_CALLBACK_EXIT – Callback

Exit providers can supply an MQ_CALLBACK_EXIT function to intercept messages and notifications sent to a message consumer. The exit is called before and after the consumer function has been called.

Syntax

Parameters

Usage notes

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa43450_

7.6.10.1. Syntax

MQ_CALLBACK_EXIT *(pExitParms, pExitContext, pHconn, ppMsgDesc, ppGetMsgOpts, ppBuffer, ppMQCBContext)*

Parent topic: MQ_CALLBACK_EXIT - Callback

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa43460_

7.6.10.2. Parameters

The MQ_CALLBACK_EXIT call has the following parameters:

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

ppHobj (PPMQHOBJ) - input/output

ppMsgDesc (PPMQMD) - input/output

ppGetMsgOpts (PPMQGMO) - input/output

ppBuffer (PPMQVOID) - input/output

ppContext (PPMCBC) - input/output

Parent topic: MQ_CALLBACK_EXIT - Callback

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23260_

7.6.10.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20790_

7.6.10.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20800_

7.6.10.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20810_

7.6.10.2.4. ppHobj (PPMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20820_

7.6.10.2.5. ppMsgDesc (PPMQMD) – input/output

Message descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21240_

7.6.10.2.6. ppGetMsgOpts (PPMQGMO) – input/output

Options that control the action of MQGET.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21250_

7.6.10.2.7. ppBuffer (PPMQVOID) – input/output

Area to contain the message data.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21270_

7.6.10.2.8. ppContext (PPMCBC) – input/output

Context data for the callback.

Parent topic: Parameters

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23290_

7.6.10.3. Usage notes

1. The Callback exit is invoked before the consumer is invoked and after the consumer's consumer function has completed. Although the MQMD and MQGMO structures are alterable, changing the values in the before exit does not redrive the retrieval of a message from the queue as the message has already been removed from the queue to be delivered to the consumer function

Parent topic: MQ_CALLBACK_EXIT - Callback

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa43480_

7.6.10.4. C invocation

```
MQ_CALLBACK_EXIT (&ExitParms, &ExitContext, &Hconn, &pMsgDesc,
                  &pGetMsgOpts, &pBuffer, &pContext);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PPMQMD     ppMsgDesc;      /* Message descriptor */
PPMQGMO    ppGetMsgOpts;   /* Options that define the operation of the consumer */
PPMQVOID   ppBuffer;       /* Area to conatin the message data */
PPMQCBC    ppContext;      /* Context data for the callback */
```

Parent topic: MQ_CALLBACK_EXIT - Callback

This build: January 26, 2011 11:56:15

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa43490_

7.6.11. MQ_CLOSE_EXIT – Close object

Exit providers can supply an MQ_CLOSE_EXIT function to intercept the MQCLOSE call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20760_

7.6.11.1. Syntax

```
MQ_CLOSE_EXIT (pExitParms, pExitContext, pHconn, ppHobj, pOptions,
  pCompCode, pReason)
```

Parent topic: MQ_CLOSE_EXIT - Close object

This build: January 26, 2011 11:56:00

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20770_

7.6.11.2. Parameters

The MQ_CLOSE_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

ppHobj (PPMQHOBJ) - input/output

pOptions (PMQLONG) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_CLOSE_EXIT - Close object

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20780_

7.6.11.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20790_

7.6.11.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20800_

7.6.11.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20810_

7.6.11.2.4. ppHobj (PPMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20820_

7.6.11.2.5. pOptions (PMQLONG) – input/output

Options that control the action of MQCLOSE.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20830_

7.6.11.2.6. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20840_

7.6.11.2.7. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20850_

7.6.11.3. C invocation

```
MQ_CLOSE_EXIT (&ExitParms, &ExitContext, &Hconn, &pHobj,
               &Options, &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PPMQHOBJ   ppHobj;         /* Object handle */
PMQLONG    pOptions;       /* Options that control the action of MQCLOSE */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_CLOSE_EXIT - Close object

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20860_

7.6.12. MQ_CMIT_EXIT – Commit changes

Exit providers can supply an MQ_CMIT_EXIT function to intercept the MQCMIT call. If the unit of work is being coordinated by an external unit-of-work manager, MQ_CMIT_EXIT is also invoked in response to the application issuing the unit-of-work manager's commit call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20870_

7.6.12.1. Syntax

MQ_CMIT_EXIT *(pExitParms, pExitContext, pHconn, pCompCode, pReason)*

Parent topic: MQ_CMIT_EXIT - Commit changes

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20880_

7.6.12.2. Parameters

The MQ_CMIT_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_CMIT_EXIT - Commit changes

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20890_

7.6.12.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20900_

7.6.12.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20910_

7.6.12.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20920_

7.6.12.2.4. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20930_

7.6.12.2.5. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20940_

7.6.12.3. C invocation

```
MQ_CMIT_EXIT (&ExitParms, &ExitContext, &Hconn,
              &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_CMIT_EXIT - Commit changes

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20950_

7.6.13. MQ_CONNX_EXIT – Connect queue manager (extended)

Exit providers can supply an MQ_CONNX_EXIT function to intercept the MQCONN and MQCONNX

calls.

- Syntax
- Parameters
- Usage notes
- C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20960_

7.6.13.1. Syntax

MQ_CONNX_EXIT *(pExitParms, pExitContext, pQMgrName, ppConnectOpts, ppHconn, pCompCode, pReason)*

Parent topic: MQ_CONNX_EXIT – Connect queue manager (extended)

This build: January 26, 2011 11:56:00

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20970_

7.6.13.2. Parameters

The MQ_CONNX_EXIT call has the following parameters.

- pExitParms (PMQAXP) – input/output
- pExitContext (PMQAXC) – input/output
- pQMgrName (PMQCHAR48) – input/output
- ppConnectOpts (PPMQCNO) – input/output
- ppHconn (PPMQHCONN) – input/output
- pCompCode (PMQLONG) – input/output
- pReason (PMQLONG) – input/output

Parent topic: MQ_CONNX_EXIT – Connect queue manager (extended)

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20980_

7.6.13.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa20990_

7.6.13.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21000_

7.6.13.2.3. pQMgrName (PMQCHAR48) – input/output

Name of queue manager.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21010_

7.6.13.2.4. ppConnectOpts (PPMQCNO) – input/output

Options that control the action of MQCONNX.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21020_

7.6.13.2.5. ppHconn (PPMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21030_

7.6.13.2.6. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21040_

7.6.13.2.7. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21050_

7.6.13.3. Usage notes

1. The MQ_CONNX_EXIT function interface described here is used for both the MQCONN call and the MQCONNX call. However, separate entry points are defined for these two calls. To intercept *both* calls, the MQXEP call must be used at least twice - once with function identifier MQXF_CONN, and again with MQXF_CONNX.
   Because the MQ_CONNX_EXIT interface is the same for MQCONN and MQCONNX, a single exit function can be used for both calls; the *Function* field in the MQAXP structure indicates which call is in progress. Alternatively, the MQXEP call can be used to register different exit functions for the two calls.
2. When a message channel agent (MCA) responds to an inbound client connection, the MCA can issue a number of MQ calls before the client state is fully known. These MQ calls result in the API exit functions being invoked with the MQAXC structure containing data relating to the MCA, and not to the client (for example, user identifier and connection name). However, once the client state is fully known, subsequent MQ calls result in the API exit functions being invoked with the appropriate client data in the MQAXC structure.
3. All MQXR_BEFORE exit functions are invoked before any parameter validation is performed by the queue manager. The parameters might therefore be invalid (including invalid pointers for the addresses of parameters).
   The MQ_CONNX_EXIT function is invoked before any authorization checks are performed by the queue manager.
4. The exit function must not change the name of the queue manager specified on the MQCONN or MQCONNX call. If the name is changed by the exit function, the results are undefined.
5. An MQXR_BEFORE exit function for the MQ_CONNX_EXIT cannot issue MQ calls other than MQXEP.

Parent topic: MQ_CONNX_EXIT - Connect queue manager (extended)

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21060_

7.6.13.4. C invocation

```
MQ_CONNX_EXIT (&ExitParms, &ExitContext, QMgrName,
               &pConnectOpts, &pHconn, &CompCode,
               &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP      pExitParms;      /* Exit parameter structure */
PMQAXC      pExitContext;    /* Exit context structure */
PMQCHAR48   pQMgrName;       /* Name of queue manager */
PPMQCNO     ppConnectOpts;   /* Options that control the action of
                                MQCONNX */
PPMQHCONN   ppHconn;         /* Connection handle */
PMQLONG     pCompCode;       /* Completion code */
PMQLONG     pReason;         /* Reason code qualifying CompCode */
```

Parent topic: MQ_CONNX_EXIT - Connect queue manager (extended)

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21070_

7.6.14. MQ_DISC_EXIT – Disconnect queue manager

Exit providers can supply an MQ_DISC_EXIT function to intercept the MQDISC call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21080_

7.6.14.1. Syntax

MQ_DISC_EXIT *(pExitParms, pExitContext, ppHconn, pCompCode, pReason)*

Parent topic: MQ_DISC_EXIT – Disconnect queue manager

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21090_

7.6.14.2. Parameters

The MQ_DISC_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

ppHconn (PPMQHCONN) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_DISC_EXIT – Disconnect queue manager

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21100_

7.6.14.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21110_

7.6.14.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21120_

7.6.14.2.3. ppHconn (PPMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21130_

7.6.14.2.4. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21140_

7.6.14.2.5. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21150_

7.6.14.3. C invocation

```
MQ_DISC_EXIT (&ExitParms, &ExitContext, &pHconn,
              &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP      pExitParms;     /* Exit parameter structure */
PMQAXC      pExitContext;   /* Exit context structure */
PPMQHCONN   ppHconn;        /* Connection handle */
PMQLONG     pCompCode;      /* Completion code */
PMQLONG     pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_DISC_EXIT - Disconnect queue manager

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21160_

7.6.15. MQ_GET_EXIT – Get message

Exit providers can supply an MQ_GET_EXIT function to intercept the MQGET call. The same exit function interface is used for the MQXF_DATA_CONV_ON_GET exit function.

Syntax

Parameters

Usage notes

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21170_

7.6.15.1. Syntax

MQ_GET_EXIT *(pExitParms, pExitContext, pHconn, pHobj, ppMsgDesc, ppGetMsgOpts, pBufferLength, ppBuffer, ppDataLength, pCompCode, pReason)*

Parent topic: MQ_GET_EXIT - Get message

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21180_

7.6.15.2. Parameters

The MQ_GET_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

pHobj (PMQHOBJ) - input/output

ppMsgDesc (PPMQMD) - input/output

ppGetMsgOpts (PPMQGMO) - input/output

pBufferLength (PMQLONG) - input/output

ppBuffer (PPMQVOID) - input/output

ppDataLength (PPMQLONG) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_GET_EXIT - Get message

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21190_

7.6.15.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21200_

7.6.15.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21210_

7.6.15.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21220_

7.6.15.2.4. pHobj (PMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21230_

7.6.15.2.5. ppMsgDesc (PPMQMD) – input/output

Message descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21240_

7.6.15.2.6. ppGetMsgOpts (PPMQGMO) – input/output

Options that control the action of MQGET.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21250_

7.6.15.2.7. pBufferLength (PMQLONG) – input/output

Length in bytes of the *ppBuffer* area.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21260_

7.6.15.2.8. ppBuffer (PPMQVOID) – input/output

Area to contain the message data.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21270_

7.6.15.2.9. ppDataLength (PPMQLONG) – input/output

Length of the message.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21280_

7.6.15.2.10. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21290_

7.6.15.2.11. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21300_

7.6.15.3. Usage notes

1. The MQ_GET_EXIT function interface described here is used for both the MQXF_GET exit function and the MQXF_DATA_CONV_ON_GET exit function. However, separate entry points are defined for these two exit functions, so to intercept *both* the MQXEP call must be used twice - once with function identifier MQXF_GET, and again with MQXF_DATA_CONV_ON_GET. Because the MQ_GET_EXIT interface is the same for MQXF_GET and MQXF_DATA_CONV_ON_GET, a single exit function can be used for both; the *Function* field in the MQAXP structure indicates which exit function has been invoked. Alternatively, the MQXEP call can be used to register different exit functions for the two cases.
2. There is no MQXR_AFTER exit function for MQXF_DATA_CONV_ON_GET; the MQXR_AFTER exit function for MQXF_GET provides the required capability for exit processing after data conversion.

Parent topic: MQ_GET_EXIT - Get message

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21310_

7.6.15.4. C invocation

```
MQ_GET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj,
             &pMsgDesc, &pGetMsgOpts, &BufferLength,
             &pBuffer, &pDataLength, &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;      /* Exit parameter structure */
PMQAXC     pExitContext;    /* Exit context structure */
PMQHCONN   pHconn;          /* Connection handle */
PMQHOBJ    pHobj;           /* Object handle */
PPMQMD     ppMsgDesc;       /* Message descriptor */
PPMQGMO    ppGetMsgOpts;    /* Options that control the action of MQGET */
PMQLONG    pBufferLength;   /* Length in bytes of the pBuffer area */
PPMQVOID   ppBuffer;        /* Area to contain the message data */
PPMQLONG   ppDataLength;    /* Length of the message */
PMQLONG    pCompCode;       /* Completion code */
PMQLONG    pReason;         /* Reason code qualifying CompCode */
```

Parent topic: MQ_GET_EXIT - Get message

This build: January 26, 2011 11:56:01

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21320_

7.6.16. MQ_INIT_EXIT – Initialize exit environment

Exit providers can supply an MQ_INIT_EXIT function to perform connection-level initialization.

Syntax

Parameters

Usage notes

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21330_

7.6.16.1. Syntax

MQ_INIT_EXIT *(pExitParms, pExitContext, pCompCode, pReason)*

Parent topic: MQ_INIT_EXIT – Initialize exit environment

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21340_

7.6.16.2. Parameters

The MQ_INIT_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_INIT_EXIT – Initialize exit environment

This build: January 26, 2011 11:56:01

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21350_

7.6.16.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21360_

7.6.16.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21370_

7.6.16.2.3. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21380_

7.6.16.2.4. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21390_

7.6.16.3. Usage notes

1. The MQ_INIT_EXIT function can issue the MQXEP call to register the addresses of the exit functions for the particular MQ calls to be intercepted. It is not necessary to intercept all MQ calls, or to intercept both MQXR_BEFORE and MQXR_AFTER calls. For example, an exit suite could choose to intercept only the MQXR_BEFORE call of MQPUT.
2. Storage that is to be used by exit functions in the exit suite can be acquired by the MQ_INIT_EXIT function. Alternatively, exit functions can acquire storage when they are invoked, as and when needed. However, all storage should be freed before the exit suite is terminated; the MQ_TERM_EXIT function can free the storage, or an exit function invoked earlier.
3. If MQ_INIT_EXIT returns MQXCC_FAILED in the *ExitResponse* field of MQAXP, or fails in some other way, the MQCONN or MQCONNX call that caused MQ_INIT_EXIT to be invoked also fails, with the *CompCode* and *Reason* parameters set to appropriate values.
4. An MQ_INIT_EXIT function cannot issue MQ calls other than MQXEP.

Parent topic: MQ_INIT_EXIT - Initialize exit environment

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21400_

7.6.16.4. C invocation

```
MQ_INIT_EXIT (&ExitParms, &ExitContext, &CompCode,
              &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP    pExitParms;     /* Exit parameter structure */
```

```
PMQAXC   pExitContext;  /* Exit context structure */
PMQLONG  pCompCode;     /* Completion code */
PMQLONG  pReason;       /* Reason code qualifying CompCode */
```

Parent topic: MQ_INIT_EXIT - Initialize exit environment

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21410_

7.6.17. MQ_INQ_EXIT – Inquire object attributes

Exit providers can supply an MQ_INQ_EXIT function to intercept the MQINQ call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21420_

7.6.17.1. Syntax

MQ_INQ_EXIT *(pExitParms, pExitContext, pHconn, pHobj, pSelectorCount, ppSelectors, pIntAttrCount, ppIntAttrs, pCharAttrLength, ppCharAttrs, pCompCode, pReason)*

Parent topic: MQ_INQ_EXIT - Inquire object attributes

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21430_

7.6.17.2. Parameters

The MQ_INQ_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

pHobj (PMQHOBJ) - input/output

pSelectorCount (PMQLONG) - input/output

ppSelectors (PPMQLONG) - input/output

pIntAttrCount (PMQLONG) - input/output

ppIntAttrs (PPMQLONG) - input/output

pCharAttrLength (PMQLONG) - input/output

ppCharAttrs (PPMQCHAR) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_INQ_EXIT - Inquire object attributes

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21440_

7.6.17.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21450_

7.6.17.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21460_

7.6.17.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21470_

7.6.17.2.4. pHobj (PMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21480_

7.6.17.2.5. pSelectorCount (PMQLONG) – input/output

Count of selectors.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21490_

7.6.17.2.6. ppSelectors (PPMQLONG) – input/output

Array of attribute selectors.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21500_

7.6.17.2.7. pIntAttrCount (PMQLONG) – input/output

Count of integer attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21510_

7.6.17.2.8. ppIntAttrs (PPMQLONG) – input/output

Array of integer attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21520_

7.6.17.2.9. pCharAttrLength (PMQLONG) – input/output

Length of character attributes buffer.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21530_

7.6.17.2.10. ppCharAttrs (PPMQCHAR) – input/output

Character attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa21540_

7.6.17.2.11. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21550_

7.6.17.2.12. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21560_

7.6.17.3. C invocation

```
MQ_INQ_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj,
             &SelectorCount, &pSelectors, &IntAttrCount,
             &pIntAttrs, &CharAttrLength, &pCharAttrs,
             &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;        /* Exit parameter structure */
PMQAXC     pExitContext;      /* Exit context structure */
PMQHCONN   pHconn;            /* Connection handle */
PMQHOBJ    pHobj;             /* Object handle */
PMQLONG    pSelectorCount;    /* Count of selectors */
PPMQLONG   ppSelectors;       /* Array of attribute selectors */
PMQLONG    pIntAttrCount;     /* Count of integer attributes */
PPMQLONG   ppIntAttrs;        /* Array of integer attributes */
PMQLONG    pCharAttrLength;   /* Length of character attributes buffer */
PPMQCHAR   ppCharAttrs;       /* Character attributes */
PMQLONG    pCompCode;         /* Completion code */
PMQLONG    pReason;           /* Reason code qualifying CompCode */
```

Parent topic: MQ_INQ_EXIT - Inquire object attributes

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21570_

7.6.18. MQ_OPEN_EXIT – Open object

Exit providers can supply an MQ_OPEN_EXIT function to intercept the MQOPEN call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21580_

7.6.18.1. Syntax

```
MQ_OPEN_EXIT (pExitParms, pExitContext, pHconn, ppObjDesc,
  pOptions, ppHobj, pCompCode, pReason)
```

Parent topic: MQ_OPEN_EXIT – Open object

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21590_

7.6.18.2. Parameters

The MQ_OPEN_EXIT call has the following parameters.

- pExitParms (PMQAXP) – input/output
- pExitContext (PMQAXC) – input/output
- pHconn (PMQHCONN) – input/output
- ppObjDesc (PPMQOD) – input/output
- pOptions (PMQLONG) – input/output
- ppHobj (PPMQHOBJ) – input/output
- pCompCode (PMQLONG) – input/output
- pReason (PMQLONG) – input/output

Parent topic: MQ_OPEN_EXIT – Open object

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21600_

7.6.18.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21610_

7.6.18.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21620_

7.6.18.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21630_

7.6.18.2.4. ppObjDesc (PPMQOD) – input/output

Object descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21640_

7.6.18.2.5. pOptions (PMQLONG) – input/output

Options that control the action of MQ_OPEN_EXIT.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21650_

7.6.18.2.6. ppHobj (PPMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21660_

7.6.18.2.7. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.

This topic's URL:
fa21670_

7.6.18.2.8. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21680_

7.6.18.3. C invocation

```
MQ_OPEN_EXIT (&ExitParms, &ExitContext, &Hconn,
              &pObjDesc, &Options, &pHobj, &CompCode,
              &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PPMQOD     ppObjDesc;      /* Object descriptor */
PMQLONG    pOptions;       /* Options that control the action of
                              MQ_OPEN_EXIT */
PPMQHOBJ   ppHobj;         /* Object handle */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_OPEN_EXIT – Open object

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21690_

7.6.19. MQ_PUT_EXIT – Put message

Exit providers can supply an MQ_PUT_EXIT function to intercept the MQPUT call.

Syntax

Parameters

Usage notes

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21700_

7.6.19.1. Syntax

MQ_PUT_EXIT *(pExitParms, pExitContext, pHconn, pHobj, ppMsgDesc, ppPutMsgOpts, pBufferLength, ppBuffer, pCompCode, pReason)*

Parent topic: MQ_PUT_EXIT – Put message

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21710_

7.6.19.2. Parameters

The MQ_PUT_EXIT call has the following parameters.

- pExitParms (PMQAXP) – input/output
- pExitContext (PMQAXC) – input/output
- pHconn (PMQHCONN) – input/output
- pHobj (PMQHOBJ) – input/output
- ppMsgDesc (PPMQMD) – input/output
- ppPutMsgOpts (PPMQPMO) – input/output
- pBufferLength (PMQLONG) – input/output
- ppBuffer (PPMQVOID) – input/output
- pCompCode (PMQLONG) – input/output
- pReason (PMQLONG) – input/output

Parent topic: MQ_PUT_EXIT – Put message

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21720_

7.6.19.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21730_

7.6.19.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21740_

7.6.19.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21750_

7.6.19.2.4. pHobj (PMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21760_

7.6.19.2.5. ppMsgDesc (PPMQMD) – input/output

Message descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:02

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21770_

7.6.19.2.6. ppPutMsgOpts (PPMQPMO) – input/output

Options that control the action of MQPUT.

Parent topic: Parameters

This build: January 26, 2011 11:56:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21780_

7.6.19.2.7. pBufferLength (PMQLONG) – input/output

Length of the message in *pBuffer*.

Parent topic: Parameters

This build: January 26, 2011 11:56:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21790_

7.6.19.2.8. ppBuffer (PPMQVOID) – input/output

Message data.

Parent topic: Parameters

This build: January 26, 2011 11:56:03

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21800_

7.6.19.2.9. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21810_

7.6.19.2.10. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21820_

7.6.19.3. Usage notes

- Report messages generated by the queue manager skip the normal call processing. As a result, such messages cannot be intercepted by the MQ_PUT_EXIT function or the MQPUT1 function. However, report messages generated by the message channel agent are processed normally, and hence can be intercepted by the MQ_PUT_EXIT function or the MQ_PUT1_EXIT function. To be sure to intercepting all of the report messages generated by the MCA, both MQ_PUT_EXIT and MQ_PUT1_EXIT should be used.

Parent topic: MQ_PUT_EXIT - Put message

This build: January 26, 2011 11:56:04

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21830_

7.6.19.4. C invocation

```
MQ_PUT_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj,
             &pMsgDesc, &pPutMsgOpts, &BufferLength,
             &pBuffer, &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;      /* Exit parameter structure */
PMQAXC     pExitContext;    /* Exit context structure */
PMQHCONN   pHconn;          /* Connection handle */
PMQHOBJ    pHobj;           /* Object handle */
PPMQMD     ppMsgDesc;       /* Message descriptor */
PPMQPMO    ppPutMsgOpts;    /* Options that control the action of MQPUT */
PMQLONG    pBufferLength;   /* Length of the message in pBuffer */
PPMQVOID   ppBuffer;        /* Message data */
PMQLONG    pCompCode;       /* Completion code */
PMQLONG    pReason;         /* Reason code qualifying CompCode */
```

Parent topic: MQ_PUT_EXIT - Put message

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21840_

7.6.20. MQ_PUT1_EXIT – Put one message

Exit providers can supply an MQ_PUT1_EXIT function to intercept the MQPUT1 call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21850_

7.6.20.1. Syntax

MQ_PUT1_EXIT *(pExitParms, pExitContext, pHconn, ppObjDesc, ppMsgDesc, ppPutMsgOpts, pBufferLength, ppBuffer, pCompCode, pReason)*

Parent topic: MQ_PUT1_EXIT – Put one message

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21860_

7.6.20.2. Parameters

The MQ_PUT1_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

pHconn (PMQHCONN) – input/output

ppObjDesc (PPMQOD) – input/output

ppMsgDesc (PPMQMD) – input/output

ppPutMsgOpts (PPMQPMO) – input/output

pBufferLength (PMQLONG) – input/output

ppBuffer (PPMQVOID) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_PUT1_EXIT – Put one message

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21870_

7.6.20.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21880_

7.6.20.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21890_

7.6.20.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21900_

7.6.20.2.4. ppObjDesc (PPMQOD) – input/output

Object descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21910_

7.6.20.2.5. ppMsgDesc (PPMQMD) – input/output

Message descriptor.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21920_

7.6.20.2.6. ppPutMsgOpts (PPMQPMO) – input/output

Options that control the action of MQPUT1.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21930_

7.6.20.2.7. pBufferLength (PMQLONG) – input/output

Length of the message in *ppBuffer*.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21940_

7.6.20.2.8. ppBuffer (PPMQVOID) – input/output

Message data.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21950_

7.6.20.2.9. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21960_

7.6.20.2.10. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21970_

7.6.20.3. C invocation

```
MQ_PUT1_EXIT (&ExitParms, &ExitContext, &Hconn,
              &pObjDesc, &pMsgDesc, &pPutMsgOpts,
              &BufferLength, &pBuffer, &CompCode,
              &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;      /* Exit parameter structure */
PMQAXC     pExitContext;    /* Exit context structure */
PMQHCONN   pHconn;          /* Connection handle */
PPMQOD     ppObjDesc;       /* Object descriptor */
PPMQMD     ppMsgDesc;       /* Message descriptor */
PPMQPMO    ppPutMsgOpts;    /* Options that control the action of MQPUT1 */
PMQLONG    pBufferLength;   /* Length of the message in pBuffer */
PPMQVOID   ppBuffer;        /* Message data */
PMQLONG    pCompCode;       /* Completion code */
PMQLONG    pReason;         /* Reason code qualifying CompCode */
```

Parent topic: MQ_PUT1_EXIT - Put one message

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21980_

7.6.21. MQ_SET_EXIT – Set object attributes

Exit providers can supply an MQ_SET_EXIT function to intercept the MQSET call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa21990_

7.6.21.1. Syntax

```
MQ_SET_EXIT (pExitParms, pExitContext, pHconn, pHobj, pSelectorCount,
  ppSelectors, pIntAttrCount, ppIntAttrs, pCharAttrLength, ppCharAttrs, pCompCode, pReason)
```

Parent topic: MQ_SET_EXIT - Set object attributes

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22000_

7.6.21.2. Parameters

The MQ_SET_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pHconn (PMQHCONN) - input/output

pHobj (PMQHOBJ) - input/output

pSelectorCount (PMQLONG) - input/output

ppSelectors (PPMQLONG) - input/output

pIntAttrCount (PMQLONG) - input/output

ppIntAttrs (PPMQLONG) - input/output

pCharAttrLength (PMQLONG) - input/output

ppCharAttrs (PPMQCHAR) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_SET_EXIT - Set object attributes

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22010_

7.6.21.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22020_

7.6.21.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22030_

7.6.21.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22040_

7.6.21.2.4. pHobj (PMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22050_

7.6.21.2.5. pSelectorCount (PMQLONG) – input/output

Count of selectors.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22060_

7.6.21.2.6. ppSelectors (PPMQLONG) – input/output

Array of attribute selectors.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22070_

7.6.21.2.7. pIntAttrCount (PMQLONG) – input/output

Count of integer attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22080_

7.6.21.2.8. ppIntAttrs (PPMQLONG) – input/output

Array of integer attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22090_

7.6.21.2.9. pCharAttrLength (PMQLONG) – input/output

Length of character attributes buffer.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22100_

7.6.21.2.10. ppCharAttrs (PPMQCHAR) – input/output

Character attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22110_

7.6.21.2.11. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22120_

7.6.21.2.12. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22130_

7.6.21.3. C invocation

```
MQ_SET_EXIT (&ExitParms, &ExitContext, &Hconn, &Hobj,
             &SelectorCount, &pSelectors, &IntAttrCount,
             &pIntAttrs, &CharAttrLength, &pCharAttrs,
             &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;        /* Exit parameter structure */
PMQAXC     pExitContext;      /* Exit context structure */
PMQHCONN   pHconn;            /* Connection handle */
PMQHOBJ    pHobj;             /* Object handle */
PMQLONG    pSelectorCount;    /* Count of selectors */
PPMQLONG   ppSelectors;       /* Array of attribute selectors */
PMQLONG    pIntAttrCount;     /* Count of integer attributes */
PPMQLONG   ppIntAttrs;        /* Array of integer attributes */
PMQLONG    pCharAttrLength;   /* Length of character attributes buffer */
PPMQCHAR   ppCharAttrs;       /* Character attributes */
PMQLONG    pCompCode;         /* Completion code */
PMQLONG    pReason;           /* Reason code qualifying CompCode */
```

Parent topic: MQ_SET_EXIT - Set object attributes

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22140_

7.6.22. MQ_SUB_EXIT – Register subscription

Exit providers can supply an MQ_SUB_EXIT function to intercept the MQSUB call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23000_

7.6.22.1. Syntax

MQ_SUB_EXIT
(pExitParms, pExitContext, pHconn, ppSubDesc, ppHobj, ppHsub, pCompCode, pReason)

Parent topic: MQ_SUB_EXIT – Register subscription

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23010_

7.6.22.2. Parameters

The MQ_SUB_EXIT call has the following parameters.

pExitParms (PMQAXP) – input/output

pExitContext (PMQAXC) – input/output

pHconn (PMQHCONN) – input/output

ppSubDesc (PPMQSD) – input/output

ppHobj (PPMQHOBJ) – input/output

ppHsub (PPMQHOBJ) – input/output

pCompCode (PMQLONG) – input/output

pReason (PMQLONG) – input/output

Parent topic: MQ_SUB_EXIT – Register subscription

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23020_

7.6.22.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23030_

7.6.22.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23040_

7.6.22.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23050_

7.6.22.2.4. ppSubDesc (PPMQSD) – input/output

Array of attribute selectors.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23060_

7.6.22.2.5. ppHobj (PPMQHOBJ) – input/output

Object handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23070_

7.6.22.2.6. ppHsub (PPMQHOBJ) – input/output

Array of integer attributes.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23080_

7.6.22.2.7. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23090_

7.6.22.2.8. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23100_

7.6.22.3. C invocation

```
MQ_SUB_EXIT (&ExitParms, &ExitContext, &Hconn, &pObjDesc, &Options,
&pHobj, &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;         /* Connection handle */
PPMQSD     ppSubDesc;      /* Subscription Descriptor */
PPMQHOBJ   ppHobj;         /* Object handle */
PPMQHOBJ   ppHsub;         /* Subscription handle */
PMQLONG    pCompCode;      /* Completion code */
PMQLONG    pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_SUB_EXIT - Register subscription

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23110_

7.6.23. MQ_SUBRQ_EXIT – Subscription request

Exit providers can supply an MQ_SUBRQ_EXIT function to intercept the MQSUBRQ call.

Syntax

Parameters

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23120_

7.6.23.1. Syntax

```
MQ_SUBRQ_EXIT
  (pExitParms, pExitContext, pHconn, ppHsub, pAction, ppSubRqOpts, pCompCode, pReason)
```

Parent topic: MQ_SUBRQ_EXIT - Subscription request

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23130_

7.6.23.2. Parameters

The MQ_SUBRQ_EXIT call has the following parameters.

- pExitParms (PMQAXP) - input/output
- pExitContext (PMQAXC) - input/output
- pHconn (PMQHCONN) - input/output
- ppHSub (PPMQHOBJ) - input/output
- pAction (PMQLONG) - input/output
- ppSubRqOpts (PPMQSRO) - input/output
- pCompCode (PMQLONG) - input/output
- pReason (PMQLONG) - input/output

Parent topic: MQ_SUBRQ_EXIT - Subscription request

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23140_

7.6.23.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23150_

7.6.23.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23160_

7.6.23.2.3. pHconn (PMQHCONN) – input/output

Connection handle.

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23170_

7.6.23.2.4. ppHSub (PPMQHOBJ) – input/output

Subscription handle

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23180_

7.6.23.2.5. pAction (PMQLONG) – input/output

Action

Parent topic: Parameters

This build: January 26, 2011 11:56:14

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23190_

7.6.23.2.6. ppSubRqOpts (PPMQSRO) – input/output

Subscription request options

Parent topic: Parameters

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23200_

7.6.23.2.7. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23210_

7.6.23.2.8. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23220_

7.6.23.3. C invocation

```
MQ_SUBRQ_EXIT (&ExitParms, &ExitContext, &Hconn, &pHsub, &Action, &pSubRqOpts,
 &CompCode, &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP     pExitParms;     /* Exit parameter structure */
PMQAXC     pExitContext;   /* Exit context structure */
PMQHCONN   pHconn;          /* Connection handle */
PPMQHOBJ   ppHsub;          /* Subscription handle */
PMQLONG    pAction;         /* Action */
PPMQSRO    ppSubRqOpts;    /* Subscription Request Options */
PMQLONG    pCompCode;       /* Completion code */
PMQLONG    pReason;         /* Reason code qualifying CompCode */
```

Parent topic: MQ_SUBRQ_EXIT – Subscription request

This build: January 26, 2011 11:56:15

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa23230_

7.6.24. MQ_TERM_EXIT – Terminate exit environment

Exit providers can supply an MQ_INIT_EXIT function to perform connection-level termination.

Syntax

Parameters

Usage notes

C invocation

Parent topic: API exit reference information

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22150_

7.6.24.1. Syntax

```
MQ_TERM_EXIT (pExitParms, pExitContext, pCompCode, pReason)
```

Parent topic: MQ_TERM_EXIT – Terminate exit environment

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22160_

7.6.24.2. Parameters

The MQ_TERM_EXIT call has the following parameters.

pExitParms (PMQAXP) - input/output

pExitContext (PMQAXC) - input/output

pCompCode (PMQLONG) - input/output

pReason (PMQLONG) - input/output

Parent topic: MQ_TERM_EXIT - Terminate exit environment

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22170_

7.6.24.2.1. pExitParms (PMQAXP) – input/output

Exit parameter structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22180_

7.6.24.2.2. pExitContext (PMQAXC) – input/output

Exit context structure.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22190_

7.6.24.2.3. pCompCode (PMQLONG) – input/output

Completion code.

Parent topic: Parameters

This build: January 26, 2011 11:56:05

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22200_

7.6.24.2.4. pReason (PMQLONG) – input/output

Reason code qualifying *pCompCode*.

Parent topic: Parameters

This build: January 26, 2011 11:56:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22210_

7.6.24.3. Usage notes

1. The MQ_TERM_EXIT function is optional. It is not necessary for an exit suite to register a termination exit if there is no termination processing to be done.
   If functions belonging to the exit suite acquire resources during the connection, an MQ_TERM_EXIT function is a convenient point at which to free those resources, for example, freeing storage obtained dynamically.
2. If an MQ_TERM_EXIT function is registered when the MQDISC call is issued, the exit function is invoked after all of the MQDISC exit functions have been invoked.
3. If MQ_TERM_EXIT returns MQXCC_FAILED in the *ExitResponse* field of MQAXP, or fails in some other way, the MQDISC call that caused MQ_TERM_EXIT to be invoked also fails, with the *CompCode* and *Reason* parameters set to appropriate values.

Parent topic: MQ_TERM_EXIT - Terminate exit environment

This build: January 26, 2011 11:56:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22220_

7.6.24.4. C invocation

```
MQ_TERM_EXIT (&ExitParms, &ExitContext, &CompCode,
              &Reason);
```

The parameters passed to the exit are declared as follows:

```
PMQAXP    pExitParms;     /* Exit parameter structure */
PMQAXC    pExitContext;   /* Exit context structure */
PMQLONG   pCompCode;      /* Completion code */
PMQLONG   pReason;        /* Reason code qualifying CompCode */
```

Parent topic: MQ_TERM_EXIT - Terminate exit environment

This build: January 26, 2011 11:56:06

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22230_

7.7. Referencing connection definitions using a pre-connect exit from a repository

WebSphere MQ clients can be configured to look up a repository to obtain connection definitions using a pre-connect exit library.

Introduction

A client application can connect to a queue manager using client channel definition tables (CCDT). Generally, the CCDT file is located on a central network file server, and have clients referencing it. Since it is difficult to manage and administer various client applications referencing the CCDT file, a flexible approach is to store the client definitions in a global repository like an LDAP directory, a WebSphere Registry and Repository or any other repository. Storing the client connection definitions in a repository makes managing client connection definitions easier, and applications can access the correct and most current client connection definitions.

During the MQCONN/X call execution, WebSphere MQ C Client loads an application specified pre-connect exit library, and invokes an exit function to retrieve connection definitions. The retrieved connection definitions are then used to establish connection to a queue manager. The details of exit library and function to invoke are specified in the mqclient.ini configuration file.

Syntax

```
void MQ_PRECONNECT_EXIT (pExitParms, pQMgrName, ppConnectOpts, pCompCode, pReason);
```

Parameters

pExitParms

Type: PMQNXP input /output

The PreConnection exit parameter structure.

The structure is allocated and maintained by the caller of the exit.

pQMgrName

Type: PMQCHAR input/output

Name of the queue manager.

On input, this parameter is the filter string supplied to the MQCONN API call through the QMgrName parameter. This field might be blank, explicit, or contain certain wildcard characters. The field is changed by the exit. The parameter is NULL when the exit is called with MQXR_TERM.

ppConnectOpts

Type: ppConnectOpts input/output

Options that control the action of MQCONNX.

This is a pointer to an MQCNO connection options structure that controls the action of the MQCONN API call. The parameter is NULL when the exit is called with MQXR_TERM. The MQI client always provides an MQCNO structure to the exit, even if it was not originally provided by the application. If an application provides an MQCNO structure, the client makes a duplicate to pass it to the exit where it is modified. The client retains the ownership of the MQCNO.

An MQCD referenced through the MQCNO takes precedence over any connection definition provided through the array. The client uses the MQCNO structure to connect to the queue manager and the others are ignored.

pCompCode

Type: PMQLONG input/output

Completion code.

Pointer to an MQLONG that receives the exits completion code. It must be one of the following values:

- MQCC_OK - Successful completion
- MQCC_WARNING - Warning (partial completion)
- MQCC_FAILED - Call failed

pReason

Type: PMQLONG input/output

Reason qualifying pCompCode.

Pointer to an MQLONG that receives the exit reason code. If the completion code is MQCC_OK, the only valid value is:

- MQRC_NONE - (0, x'000') No reason to report.

If the completion code is MQCC_FAILED or MQCC_WARNING, the exit function can set the reason code field to any valid MQRC_* value.

C Invocation

```
   void MQ_PRECONNECT_EXIT (&ExitParms, &QMgrName, &pConnectOpts, &CompCode, &Reason);
```

Parameter

```
   PMQNXP  pExitParms    /*PreConnect exit parameter structure*/
   PMQCHAR pQMgrName     /*Name of the queue manager*/
   PPMQCNO ppConnectOpts/*Options controlling the action of MQCONNX*/
   PMQLONG pCompCode     /*Completion code*/
   PMQLONG pReason       /*Reason qualifying pCompCode*/
```

Parent topic: WebSphere MQ installable services and the API exit

This build: January 26, 2011 11:56:16

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa24020_

8. File Transfer Application

This chapter explains how to use the File Transfer Application.

The chapter includes:

- Introduction
- Installing and configuring
- Using the File Transfer Application

Introduction
The File Transfer Application allows you to send and receive files in the form of WebSphere MQ messages. You can use the File Transfer Application to send and receive binary files (for example, image files, wordprocessor files, spreadsheet files, or compressed files).

Installing and configuring

Using the File Transfer Application

Parent topic: System Administration Guide

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22360_

8.1. Introduction

The File Transfer Application allows you to send and receive files in the form of WebSphere® MQ messages. You can use the File Transfer Application to send and receive binary files (for example, image files, wordprocessor files, spreadsheet files, or compressed files).

The File Transfer Application is available on both WebSphere MQ for Windows, and WebSphere MQ for Linux (x86 platform) servers, and clients.

The File Transfer Application includes a graphical user interface (GUI), and a command-line interface. Both interfaces invoke the same underlying WebSphere MQ functionality and can be used to transfer the same types of file.

There are three principal users of the File Transfer Application:

System administrators

Set up the File Transfer Application for end-users, by defining queues as sources and destinations in the File Transfer Application GUI.

Non-experienced end-users

These are non-expert IT users working within the business, for example car showroom managers. Users in this group want to send and receive files such as daily sales figures and stock reports, using the File Transfer Application GUI.

Experienced end-users

These are expert IT users working within the business, who have an understanding of WebSphere MQ. Users in this group want to send and receive files from the command-line.

Advantages

Components

Parent topic: File Transfer Application

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22370_

8.1.1. Advantages

- *Files of any type can be transferred*. Because the File Transfer Application does not distinguish between files of different types, you can send and receive files in any format (for example, spreadsheets, memos, letters). You can even send and receive image and sound files.
- *File transfer is technology independent*. Files can be transferred between dissimilar operating platforms (for example Windows, UNIX), using TCP/IP.
- *Transferred files cannot be accidentally duplicated*. Files are sent once-and-once-only to a specified destination.
- *Files are transferred securely*. High-level data security and integrity is provided if SSL (secure sockets layer) encrypted message channels are used.
- *The sender and receiver run independently*. The sender and receiver do not both have to be running at the same time.
  If the receiver is currently unavailable or busy, the file is held on a queue. When the receiver becomes available, the file is then automatically transferred. The persistent option assures maximum reliability (non-persistent messages are automatically deleted from the queue on the receiving machine, when the queue manager restarts).
- *The system is scalable*. An administrator can add new sources and destinations to the File Transfer Application GUI, so that they become accessible to users.
- *The File Transfer Application GUI*. The File Transfer Application provides a GUI.

Parent topic: Introduction

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22380_

8.1.2. Components

The File Transfer Application consists of the following main components:

Sender

This is a program that puts a file stored in the local file system onto a queue, as one or more WebSphere® MQ messages.

Receiver

This is a program that receives files and stores them in a local file system.

File Transfer Application GUI

The GUI allows non-experienced users to send files, receive files, and create a list of sent/received files in an intuitive way. Users of the GUI need no knowledge of how the underlying WebSphere MQ technology works.

Available when installed from a WebSphere MQ server CD.

A command line interface

This provides a way for experienced users to send and receive files by issuing commands from the command line. Additional functionality is available with the File Transfer Application GUI. Users of the command line interface need to have an understanding of how WebSphere MQ works.

Parent topic: Introduction

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22390_

8.2. Installing and configuring

The File Transfer Application is used to transfer files between remote machines. The File Transfer Application can be installed on a WebSphere® MQ server or on a WebSphere MQ client. For a file to be transferred between two remote machines, the File Transfer Application must be installed on both the sender machine and the receiver machine. Having installed the File Transfer Application additional setup and configuration tasks must be performed.

Installing the File Transfer Application on a WebSphere MQ server

Installing the File Transfer Application on a WebSphere MQ client

Setup tasks

Configuring the GUI

File Transfer Application channel security

Parent topic: File Transfer Application

This build: January 26, 2011 11:56:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22400_

8.2.1. Installing the File Transfer Application on a WebSphere MQ server

A WebSphere® MQ installation server CD must be used to install the File Transfer Application on a WebSphere MQ server. To install the File Transfer Application on a WebSphere MQ server, do one of the following:

- Install the File Transfer Application during an initial installation of a WebSphere MQ server.
- Add the File Transfer Application by modifying a WebSphere MQ server installation after the initial installation has taken place.

During the initial installation

Modifying the installation

Parent topic: Installing and configuring

This build: January 26, 2011 11:56:09

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22410_

8.2.1.1. During the initial installation

To install the File Transfer Application on a WebSphere® MQ server during the initial installation of WebSphere MQ, do the following:

1. Follow the WebSphere MQ custom installation instructions on the appropriate platform to the point where you specify the components that will be installed:
   - For the WebSphere MQ server installation instructions on Windows, see the WebSphere MQ for Windows Quick Beginnings.
   - For the WebSphere MQ server installation instructions on Linux (x86 platform), see the WebSphere MQ for Linux Quick Beginnings.
2. Ensure that Server File Transfer is selected.
3. Follow the WebSphere MQ server installation instructions to completion.

The File Transfer Application is now installed.

Note: The File Transfer Application command-line interface is always installed on a server.

Parent topic: Installing the File Transfer Application on a WebSphere MQ server

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22420_

8.2.1.2. Modifying the installation

To install the File Transfer Application on a WebSphere® MQ server by modifying the installation after the initial installation has taken place, do the following:

1. Follow the instructions for modifying an installation on the appropriate platform to the point where you specify the components that are to be added:
   - For the instructions for modifying an installation on Windows, see the WebSphere MQ for Windows Quick Beginnings.
   - For the instructions for modifying an installation on Linux (x86 platform), see the WebSphere MQ for Linux Quick Beginnings.
2. Ensure that Server File Transfer is selected.
3. Follow the instructions for modifying the installation to completion.

The File Transfer Application is now installed.
Parent topic: Installing the File Transfer Application on a WebSphere MQ server

This build: January 26, 2011 11:56:09

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22430_

8.2.2. Installing the File Transfer Application on a WebSphere MQ client

Either a WebSphere® MQ installation server CD, or a WebSphere MQ installation client CD, can be used to install the File Transfer Application on a WebSphere MQ client.

Note: If you require the File Transfer Application GUI, use a WebSphere MQ installation server CD.

To install the File Transfer Application on a WebSphere MQ client, do one of the following:

- Install the File Transfer Application during an initial installation of a WebSphere MQ client.
- Add the File Transfer Application by modifying an installation after the initial installation has taken place.

During the initial installation

Modifying the installation

Parent topic: Installing and configuring

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22440_

8.2.2.1. During the initial installation

To install the File Transfer Application on a WebSphere® MQ client during the initial installation, do the following:

1. Follow the WebSphere MQ custom installation instructions on the appropriate platform to

the point where you specify the components that will be installed:

- For the WebSphere MQ client installation instructions on Windows, see the WebSphere MQ for Windows Quick Beginnings.
- For the WebSphere MQ client installation instructions on Linux (x86 platform), see the WebSphere MQ for Linux Quick Beginnings.

2. Ensure that Client File Transfer is selected.
3. Follow the WebSphere MQ installation instructions to completion.

The File Transfer Application is now installed.

Note: »The File Transfer Application command-line interface is always installed on a client.«

Parent topic: Installing the File Transfer Application on a WebSphere MQ client

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22450_

8.2.2.2. Modifying the installation

To install the File Transfer Application on a WebSphere® MQ client by modifying the installation after the initial installation has taken place, do the following:

1. Follow the instructions for modifying an installation on the appropriate platform to the point where you specify the components that are to be added:
   - For the instructions for modifying an installation on Windows, see the WebSphere MQ for Windows Quick Beginnings.
   - For the instructions for modifying an installation on Linux (x86 platform), see the WebSphere MQ for Linux Quick Beginnings.
2. Ensure that Client File Transfer is selected.
3. Follow the instructions for modifying an installations to completion.

The File Transfer Application is now installed
Parent topic: Installing the File Transfer Application on a WebSphere MQ client

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22460_

8.2.3. Setup tasks

The setup tasks are required when the File Transfer Application is used to send files between a queue manager and a remote WebSphere® MQ client, or between two remote queue managers. If files are sent between local queue managers or clients, the following setup tasks are not required.

Sending files between remote queue managers

Sending files between a queue manager and a remote WebSphere MQ client

Parent topic: Installing and configuring

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

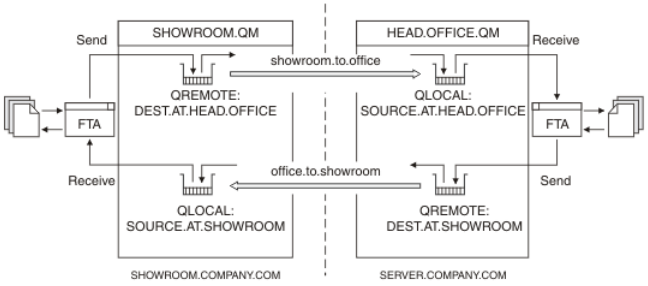
© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22470_

8.2.3.1. Sending files between remote queue managers

You can set up two queue managers to allow the File Transfer Application to send and receive files between them.

In the following illustration summarized in Figure 1, queue managers HEAD.OFFICE.QM, and SHOWROOM.QM are used.

Figure 1. Using the File Transfer Application to send files between remote queue managers



These steps outline how to setup sender and receiver channels. However, you can use other channel configurations. See WebSphere MQ Intercommunication.

1. Issue the following commands on the queue manager HEAD.OFFICE.QM to create the channels, listener, and the transmission queue:
   a. Define the sender channel:

```
DEFINE CHANNEL (OFFICE.TO.SHOWROOM) +
        CHLTYPE(SDR) +
        CONNAME (SHOWROOM.COMPANY.COM) +
        XMITQ (SHOWROOM.QM) +
        TRPTYPE(TCP)
```

   b. Define the receiver channel:

```
DEFINE CHANNEL (SHOWROOM.TO.OFFICE) +
        CHLTYPE(RCVR) +
        TRPTYPE(TCP)
```

   c. Define the listener:

```
DEFINE LISTENER (HEAD.OFFICE) +
        TRPTYPE (TCP) +
        PORT (1414)
```

   d. Define the transmission queue:

```
DEFINE QLOCAL (SHOWROOM.QM) +
        USAGE (XMITQ)
```

   Note:
   a. The TCP/IP connection names specified for the CONNAME attribute in the sender channel definitions are for illustration only. This is the network name of the machine at the *other* end of the connection. Use the values appropriate for your network.
   b. Sender and receiver channels have been used. However other channel configurations are available. See the WebSphere MQ Intercommunication manual.

2. Issue the following commands on the queue manager SHOWROOM.QM to create the channels, listener, and the transmission queue:
   a. Define the sender channel:

```
DEFINE CHANNEL (SHOWROOM.TO.OFFICE) +
        CHLTYPE(SDR) +
        CONNAME (SERVER.COMPANY.COM) +
        XMITQ (HEAD.OFFICE.QM) +
        TRPTYPE(TCP)
```

   b. Define the receiver channel:

```
DEFINE CHANNEL (OFFICE.TO.SHOWROOM) +
        CHLTYPE(RCVR) +
        TRPTYPE(TCP)
```

   c. Define the listener:

```
DEFINE LISTENER (SHOWROOM) +
        TRPTYPE (TCP) +
        PORT (1414)
```

d. Define the transmission queue:

```
DEFINE QLOCAL (HEAD.OFFICE.QM) +
       USAGE (XMITQ)
```

3. Start the listener and the sender channel on the queue manager HEAD.OFFICE.QM by using the following MQSC commands:
   a. Start the listener:

```
START LISTENER (HEAD.OFFICE)
```

   b. Start the sender channel:

```
START CHANNEL (OFFICE.TO.SHOWROOM)
```

   Note: The receiver channels do not need to be started because it is the sender channels that initiate the delivery of messages.
4. Start the listener and the sender channel on the queue manager SHOWROOM.QM by using the following MQSC commands:
   a. Start the listener:

```
START LISTENER (SHOWROOM)
```

   b. Start the sender channel:

```
START CHANNEL (SHOWROOM.TO.OFFICE)
```

5. Define a remote queue definition for the destination queue and a source queue on the queue manager HEAD.OFFICE.QM, using the following MQSC commands:
   a. Define a remote queue definition for the destination queue (the queue where files will be sent):

```
DEFINE QREMOTE (DEST.AT.SHOWROOM) +
       RNAME (SOURCE.AT.SHOWROOM) +
       RQMNAME (SHOWROOM.QM)
```

   b. Define a source queue (the queue where files will be received):

```
DEFINE QLOCAL (SOURCE.AT.HEAD.OFFICE)
```

   To avoid potential problems, ensure that local queues are dedicated to the File Transfer Application.
6. Define a remote queue definition for the destination queue and a source queue on the queue manager SHOWROOM.QM, using the following MQSC commands:
   a. Define a remote queue definition for the destination queue (the queue where files will be sent):

```
DEFINE QREMOTE (DEST.AT.HEAD.OFFICE) +
       RNAME (SOURCE.AT.HEAD.OFFICE) +
       RQMNAME (HEAD.OFFICE.QM)
```

   b. Define a source queue (the queue where files will be received):

```
DEFINE QLOCAL (SOURCE.AT.SHOWROOM)
```

   To avoid potential problems, ensure that local queues are dedicated to the File Transfer Application.
7. Ensure that all the users of the File Transfer Application are members of the mqm group, or, on Windows platforms, ensure all users are members of the local Administrators group .

You have now setup both queue managers for use with the File Transfer Application.

Parent topic: Setup tasks

This build: January 26, 2011 11:56:10

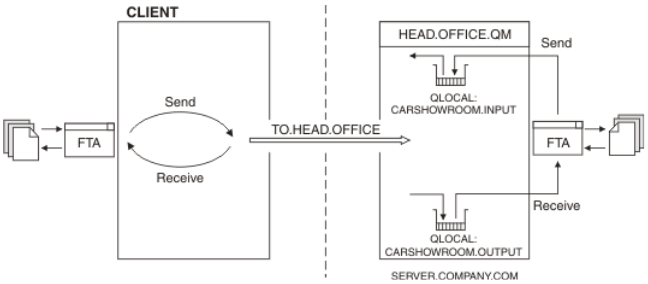
Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22480_

8.2.3.2. Sending files between a queue manager and a remote WebSphere MQ client

This section shows how to setup a queue manager and a remote WebSphere® MQ client to allow the File Transfer Application to be used to send and receive files between them. For illustration, the queue manager HEAD.OFFICE.QM, and the WebSphere MQ client CARSHOWROOM are used. Figure 1 summarizes the configuration that the following instructions form.

Figure 1. Using the File Transfer Application to send files between a queue manager and a remote client



To configure the queue manager HEAD.OFFICE.QM, and the remote WebSphere MQ client, do the following:

1. Define a server communication channel on the queue manager HEAD.OFFICE.QM, using the following MQSC command:
   a. Define a server communication channel:

```
DEFINE CHANNEL (TO.HEAD.OFFICE) +
       CHLTYPE(SVRCONN) +
       TRPTYPE(TCP) +
       MCAUSER (string)
```

   Note: For MCAUSER (*string*), specify *string* as a user from the mqm group, or administrators group on the queue manager HEAD.OFFICE.

2. Define and start a listener on the queue manager HEAD.OFFICE.QM, using the following MQSC command:
   a. Define a listener:

```
DEFINE LISTENER (HEAD.OFFICE) +
       TRPTYPE (TCP) +
       PORT (1414)
```

   b. Start the listener:

```
START LISTENER (HEAD.OFFICE)
```

3. Define a source queue, and a destination queue on the queue manager HEAD.OFFICE.QM to be used by the WebSphere MQ client, using the following MQSC commands:
   a. Define a destination queue (the queue where the WebSphere MQ client will send files):

```
DEFINE QLOCAL (CARSHOWROOM.OUTPUT)
```

   b. Define a source queue (the queue from which the WebSphere MQ client will receive files):

```
DEFINE QLOCAL (CARSHOWROOM.INPUT)
```

   It is recommended that local queues are dedicated to the File Transfer Application.

4. On the WebSphere MQ client, create an MQI channel by defining the MQSERVER environment variable as follows:

```
TO.HEAD.OFFICE/TCP/SERVER.COMPANY.COM(1414)
```

   For more information on specifying the environment variable MQSERVER, see the WebSphere MQ Clients book.
   Note: If you intend to implement SSL security, you must establish the MQI channel using a client channel definition table, and not by specifying the environment variable MQSERVER. For more information on establishing MQI channels, see the WebSphere MQ Clients book.

5. Ensure that all the users of the File Transfer Application are members of the mqm group, or alternatively the local Administrators group on Windows.

Parent topic: Setup tasks

This build: January 26, 2011 11:56:10

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22490_

8.2.4. Configuring the GUI

For every WebSphere® MQ server or client system on which you intend to run the File Transfer Application GUI, you must perform some initial configuration tasks on the GUI. Before you can configure the File Transfer Application GUI, you must have setup up your system, see Setup tasks.

For information on what the File Transfer Application GUI provides, see Components.

To configure the File Transfer Application GUI, do the following:

1. Start the File Transfer Application by issuing the control command mqftapp, or by selecting it through the start menu.
   The first time you start File Transfer Application, a message prompts you to complete the initial setup.
2. Click OK. The Queue Manager panel is displayed.
3. In the Name field, type the name of the queue manager to which the File Transfer Application is connecting.
   Note: Ensure the queue manager is running.
4. Select either the Local or Remote radio button.
   If connecting to a local queue manager, click the Local radio button.
   If connecting to a remote queue manager, click the Remote radio button.
5. Click OK. The File Transfer panel is displayed, showing a list of queues.
6. Check the tick box next to every queue that is to be used as a destination queue.
7. Select the Sources tab.
8. Check the tick box next to every queue that is to be used as a source queue.
9. Click OK.

The File Transfer Application GUI is now configured.
Parent topic: Installing and configuring

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22500_

8.2.5. File Transfer Application channel security

When using the File Transfer Application, a file is transferred over a message channel or an MQI channel. To ensure channels are secure you can use the Secure Sockets Layer (SSL), or channel exits. For information on channel security, see Channel security.

Parent topic: Installing and configuring

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22510_

8.3. Using the File Transfer Application

Once a system administrator has set up the File Transfer Application for use, files can be sent and received using WebSphere® MQ messaging. This section explains how a sender of a file, or receiver of a file, uses the File Transfer Application.

- Sending a file
- Receiving a file
- Listing all sent and received files
- File status
- Using the command line

Parent topic: File Transfer Application

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22520_

8.3.1. Sending a file

This task shows you how to send a file to another destination using the File Transfer Application GUI. You can also send files from the command line, or shell, by using mqftsnd on a WebSphere® MQ server, or mqftsndc on a WebSphere MQ client, see Using the command line.

1. Start the File Transfer Application by issuing the control command mqftapp, or by selecting it through the start menu.
2. Click Send.
3. Click Browse.
4. In the browser dialog, select the file to transfer, then click OK.
5. In the Comments field, type any accompanying comments. This field can be used to help identify different versions of the same file, for example: Week 3 sales - version 2.
6. In the Destination pane, click the required destination, for example: DESTINATION.AT.HEAD.OFFICE.
7. Click Send.
8. Look for confirmation that the file was sent in the Show files sent and received list.

The File Transfer Application transfers the specified file to the selected location. You can check that the file was sent by looking in the Session log.
Parent topic: Using the File Transfer Application

This build: January 26, 2011 11:56:10

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22530_

8.3.2. Receiving a file

This task shows how to receive a file from another destination using the File Transfer Application GUI. You can also receive files from the command line, or shell, by using mqftrcv on a WebSphere® MQ server, or mqftrcvc on a WebSphere MQ client, see Using the command line.

1. Start the File Transfer Application by issuing the control command mqftapp, or by selecting it through the start menu.
2. Click the Receive tab.
3. Click the Files from drop-down list to display the source where the file to be received is held, for example: SOURCE.AT.SHOWROOM.
4. Select the source to display the files stored there.
5. Select the file to receive, for example: Stocks 14 Aug 03.doc.
6. Look for confirmation that the file was received in the Show files sent and received list.

The selected file is received, and an icon representing the file is automatically displayed on the desktop. Double-click the icon to view the file. You can also check that the file was received by looking in the Session log.
Parent topic: Using the File Transfer Application

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22540_

8.3.3. Listing all sent and received files

This task shows how to create a list of all files sent and received using the File Transfer

Application GUI.

1. Start the File Transfer Application by issuing the control command mqftapp, or by selecting it through the start menu.
2. In the Session Log pane, click History Log....
3. View the list of all files sent or received. The list shows all files sent or received since the File Transfer Application was first used, or since the log was last cleared.
4. Click Save as.
5. Navigate to a folder on the hard drive (c:) on your machine where you want to save the list.
6. In the File name field, type a meaningful file name, for example: File log 9 Apr 03.
7. In the Save as type field, type .txt.
8. Click Save.
9. Click Done.

Parent topic: Using the File Transfer Application

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22550_

8.3.4. File status

When a File Transfer Application receiver receives files, they can be listed using the File Transfer Application GUI, or by using either of the receive file control commands. For information on the receive file control commands see mqftrcv, and mqftrcvc. Information about files in the source queue are returned as one of the following:

Complete file

When a file is too large to be transferred as a single WebSphere® MQ message, the file is segmented into a number of smaller messages, known as segments. These segments are then transmitted.

A complete file is a file where every message that forms the file has been transferred to the destination queue.

Incomplete file

An incomplete file, is a file where a subset of the messages that form the file have been transferred to the queue. If you are using the File Transfer Application GUI, by clicking Receive again, you will see the file's percentage complete figure increase.

Other message

Other messages, are messages that have not been sent using the File Transfer Application. They are not related to a file.

If a dead letter queue has been associated with the queue manager and a receive file command is issued, then other messages are put on the dead letter queue. If a dead letter queue has not been associated with the queue manager, then other messages are left on the queue.

Parent topic: Using the File Transfer Application

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22560_

8.3.5. Using the command line

This section describes how to use the File Transfer Application commands directly, without the use of the File Transfer Application GUI. If you want to use the File Transfer Application, your user ID must be a member of the mqm group. For more information about this, see Setup tasks. In addition, note the following environment-specific information:

WebSphere® MQ for Windows

All File Transfer Application commands can be issued from a command line. Command names and

their flags are not case sensitive: you can enter them in uppercase, lowercase, or a combination of uppercase and lowercase. However, arguments to specify objects (such as queue names) are case sensitive. In the syntax descriptions, the hyphen (-) is used as a flag indicator. You can use the forward slash (/) instead of the hyphen.

WebSphere MQ for Linux (x86 platform)

All File Transfer Application commands can be issued from a shell. All commands are case-sensitive.

The control commands available with the File Transfer Application follow:

Table 1. File Transfer Application command files

Command name	Purpose
mqftapp	Run the File Transfer Application GUI. For more information, see mqftapp.
mqftsnd	Send a file from a WebSphere MQ server. For more information, see mqftsnd.
mqftrcv	Receive a file on a WebSphere MQ server. For more information, see mqftrcv.
mqftsndc	Send a file from a WebSphere MQ client. For more information, see mqftsndc.
mqftrcvc	Receive a file on a WebSphere MQ client. For more information, see mqftrcvc.

The command line interface, sender, and receiver components are included as part of the WebSphere MQ clients for Windows and Linux (x86 platform).

Parent topic: Using the File Transfer Application

This build: January 26, 2011 11:56:11

Notices | Trademarks | Downloads | Library | Support | Feedback

© Copyright IBM Corporation 1999, 2009. All Rights Reserved.
This topic's URL:
fa22570_